Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Electrónica y Automática Industrial

Trabajo Fin de Grado

Automatización de un vehículo eléctrico

Autor: Javier Araluce Ruiz

Tutor/es: Luis Miguel Bergasa Pascual

2018

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

Grado en Ingeniería Electrónica y Automática Industrial

Trabajo Fin de Grado
Automatización de un vehículo eléctrico

Autor: Javier Araluce Ruiz

Tutor: Luis Miguel Bergasa Pascual

TRIBUNAL:

Presidente: Pedro Alfonso Revenga de Toro

Vocal 1º: César Mataix Gómez

Vocal 2º Luis Miguel Bergasa Pascual

FECHA: 17 de Julio de 2018

Índice general

Índice de figuras

Índice de ecuaciones

Índice de tablas

Acrónimos

TFG: Trabajo Final de Grado

GPRS: General Packet Radio Service

ADC: Analog Digital Converter

DAC: Digital Analag Converter

PCB: Printed Circuit Board

ADAS: Advanced Driver-Assistance System

LIDAR: Laser Imaging Detection and Ranging

IMU: Inertial Measurement Unit

GPS: Global Positioning System

ROS: Robot Operating System

I_2C: Inter-Integrated Circuit

RC: Resistencia y Condensador

NAND: NOT AND

GND: Ground

VCC: Voltage Common Collector

PWM: Pulse Width Modulation

GPIO: General Purpose Input/Output

USB: Universal Serial Bus

CMOS: Complimentary Metal Oxide Semiconductor

ECU: Engine Control Unit

PID: Proportional Integral Derivative

Conceptos

Bus CAN: protocolo de comunicaciones

FreeRTOS: sistema operativo en tiempo real utilizado en el micro controlador.

KMZ41: sensor de ángulo.

ADS1115: ADC de 16 bits.

DG419DY: relé de estado sólido.

CD4093BPW: dispositivo con cuatro puertas NAND.

Mapa de Lanelets: mapa realizado para la conducción autónoma del vehículo, delimitando los carriles por los que puede transitar.

OpenSource: sistema de código libre, en el que todo el mundo cualquier usuario puede adquirir el sistema y modificarlo.

Biestable JK: dispositivo capaz de mantener la salida durante un tiempo indefinido.

Inhibits: señal de activación del actuador.

Switch: interruptor.

Pull-up: resistencia a VCC para forzar el nivel alto en circuito abierto.

Pull-down: resistencia a GND para forzar el nivel bajo en circuito abierto.

Driver: controlador del motor mediante la señal de PWM y la de habilitación.

Arduino Shield: dispositivo de ampliación del micro controlador Arduino, permitiendo la expansión de sus módulos.

Gliches: características no previstas.

Resumen corto

Este trabajo presenta las tareas realizadas para la automatización de un vehículo eléctrico.

Consta de dos capítulos principales. El primero detalla las modificaciones mecánicas y los sistemas implementados para el control de la dirección y el acelerador del vehículo. Para ello se hace uso de un micro controlador que será el encargado de realizar los cálculos, empleando los datos obtenidos de los sensores añadidos al vehículo (ángulo de giro del volante desde su posición inicial y velocidad lineal del automóvil) para lograr el movimiento deseado transmitido desde el control de alto nivel.

El segundo capítulo presenta el diseño y fabricación del sistema de conmutación entre conducción manual y automática permitiendo la intervención del conductor en todo momento, otorgándole el control completo del vehículo cuando realiza cualquier acción sobre el freno, acelerador o volante. Todo ello se ha expuesto como una guía para el diseño de PCB.

Abstract

This work presents the task performed to automate an electric vehicle.

It contains two main chapters. The first one, details the mechanical modifications and systems implemented to control the steering and the accelerator of the vehicle. For that reason, it is used a micro controller to realise the calculations, using the data obtained from the sensors added to the vehicle (turning angle of the steering wheel from the initial position and the speed of the car) to achieve the movement required for the high level control.

The second chapter presents the design and manufacture of the switching system between manual and automatic driving, allowing the driver to take control of the vehicle all the time when he performs any action on the brake, Accelerator or steering wheel. This chapter has been exposed as a guide for the design of PCB.

Resumen largo

Con el inicio de la cuarta revolución industrial que está teniendo lugar y la necesidad de conectar todos los sistemas entre sí para un mejor funcionamiento y un mayor grado de autonomía**[1]** respecto al ser humano, los principales fabricantes de vehículos del mundo se encuentran con que el siguiente paso a desarrollar en esta industria se dirige hacia la autonomía completa en un vehículo. Para ello, será necesaria la integración y comunicación de diferentes sistemas dentro de éste, de manera que se consiga su completa autonomía respecto al conductor.

Para la realización de esta tarea el Instituto Federal de Investigación de Carreteras alemán (BAST)**[2]** estableció en 2013 cinco niveles de conducción autónoma como hitos intermedios hasta alcanzar la autonomía completa. Actualmente los principales fabricantes se encuentran en el tercer nivel de automatización.

Es en este contexto donde se ubica el presente trabajo de final de grado, enfocado al desarrollo hardware y a la automatización de un vehículo eléctrico, proporcionando lo que se conoce como un sistema *drive-by-wire* -que pueda ser conducido directamente enviando comandos de alto nivel-.

Las tareas realizadas se pueden dividir en tres hitos principales:

1. Modificación del **hardware** del vehículo para implementar un sistema *drive-by-wire* que posibilite la conducción autónoma. Se ha sustituido la dirección manual por una servodirección que, al disponer de un motor, puede ser controlada por un micro controlador mediante la intervención de un Driver. Además, se han medido las señales proporcionadas por el pedal del acelerador que eran comunicadas al motor, permitiendo la multiplexación mediante el módulo de conmutación para poder generar las mismas señales desde un micro controlador, logrando así controlar tanto la dirección como el acelerador del vehículo.
2. **Software** de control del vehículo. Se ha utilizado el micro controlador Olimexino-STM32**[3],** basado en Arduino, para el procesamiento de las señales. Esto ha permitido realizar un control óptimo del vehículo según las directrices otorgadas desde el control a alto nivel, que será el encargado de suministrar las órdenes al vehículo cuando éste se encuentre en modo de conducción autónoma.

3. Módulo de **conmutación** entre los dos modos de conducción -autónomo y manual-, mediante un pulsador, y aportando un elemento de seguridad que actúa sobre el volante y los pedales. Para la realización del PCB de este módulo se ha utilizado el software de código libre Kicad**[4]**, cuyo modelo ha sido enviado a una empresa de fabricación de PCBs para el posterior montaje de todos los dispositivos. La realización de esta parte del trabajo se ha expuesto como una guía para el montaje de PCBs para que otros alumnos puedan basarse en ella para la fabricación de sus prototipos.

Capítulo 1: Introducción y estado del arte

1.1 Introducción al proyecto

El grupo RobeSafe**[5]**, fundado en el año 1996, se dedica a la investigación en interfaces humano-robot, la percepción multisensorial, la localización, la navegación, SLAM y los servicios de teleasistencia. En la última década su trabajo se ha centrado en los sistemas avanzados de ayuda al conductor (ADAS) en cuanto a la toma de decisiones, proporcionando señales de posibles situaciones peligrosas en su entorno, y tomando medidas para la reducción de accidentes. El presente trabajo de final de grado se ha desarrollado dentro de este grupo de investigación que trabaja en el proyecto SmartElderlyCar**[6]** desde hace tres años.

El proyecto tiene como objetivo crear un coche autónomo orientado principalmente a usuarios de avanzada edad, debido a que algunos estudios muestran que los accidentes fatales aumentan en personas de más de 65 años. Este dato se debe a que en dicho segmento de la población a menudo se han perdido las habilidades para poder llevar a cabo alguna de las maniobras que implica la conducción. Con su automatización se lograría su realización de forma.

1.2 Estado del arte

1.2.1 Qué es un coche autónomo

Para la definición de un coche autónomo es necesario referirse al nivel de autonomía del vehículo, siendo el nivel máximo aquel en el que el vehículo es capaz de llevar a cabo todas las actividades que pueda realizar un ser humano, sin la presencia de éste. ¿Cuántos niveles de autonomía hay antes de llegar a este estado y en qué punto nos encontramos? Según la BAST:

- Nivel 0: no automatización.
  El conductor debe realizar continuamente las tareas de movimiento longitudinal (acelerar y frenar) y lateral (control del volante).

- Nivel 1: asistente de conducción.
  El conductor debe realizar continuamente las tareas de movimiento longitudinal y lateral. El resto de tareas son realizadas por el sistema dentro de unos límites. El conductor debe monitorear el sistema permanentemente y estar preparado para asumir el control total del vehículo en cualquier momento.
  Incorpora el control de crucero adaptativo y el sistema de asistencia al aparcamiento.

- Nivel 2: Automatización parcial.
  El sistema asume el control lateral y longitudinal durante un cierto período de tiempo o en situaciones específicas. El conductor debe monitorear el sistema permanentemente y estar preparado para asumir el control total del vehículo en cualquier momento.
  Incorpora el asistente de autopistas.

- Nivel 3: alta automatización
  El sistema toma el control lateral y longitudinal durante un cierto período de tiempo en situaciones específicas. El conductor no necesita monitorear el sistema permanentemente. De ser necesario, se le pedirá al conductor que se haga cargo del control, con un tiempo de

entrega suficiente. Todos los límites del sistema son conocidos por éste. No es capaz de restablecer la condición de riesgo mínimo de cada estado inicial.

Incorpora el chofer de autopista.

- Nivel 4: alto nivel de autonomía.

  El sistema toma el control lateral y longitudinal por completo. El conductor no necesita monitorear el sistema. Antes de alcanzar los límites del sistema, se le solicitará al conductor que tome el control con suficiente tiempo de antelación. En ausencia de toma de control por el conductor, el sistema volverá a la condición de riesgo mínimo. Todos los límites del sistema son conocidos por éste. Es capaz de regresar a la condición de riesgo mínimo en cada situación.

  Incorpora el piloto de autopista.

- Nivel 5: autonomía completa.

  El sistema toma el control lateral y longitudinal por completo. El conductor no necesita monitorear el sistema. El sistema no tiene límites, por lo que el vehículo podría seguir conduciendo en todo momento o circunstancia. En ausencia de toma de control por el conductor, el sistema volverá a la condición de riesgo mínimo. Todos los límites del sistema son conocidos por éste. Es capaz de regresar a la condición de riesgo mínimo en cada situación.

Para lograr la autonomía completa debe haber una interacción entre dos niveles de control: alto y bajo. Dentro del primero queda englobado el control de los sensores que se encargan de dirigir el vehículo, detectando primero el entorno para realizar la conducción autónoma. Una vez realizado el control de cómo debe ser el movimiento, es el momento de hacerlo posible. Aquí es donde entra en juego el control a bajo nivel, que es el foco principal de esta investigación.

Antes de centrar la exposición en el trabajo realizado, es necesario hacer una breve presentación de los sensores que se manejan en esta parte del control, junto a una aproximación a la articulación de esta fase, imprescindible para el funcionamiento del vehículo.

Los sensores y sistemas utilizados son:

- **LIDAR.**

  Sensor situado en el techo del vehículo. Permite conocer la distancia desde un emisor láser a un objeto o superficie utilizando un haz de pulsos láser. Se determina la distancia al objeto conociendo el tiempo que tarda en ser reflejado el pulso láser, recreando así una imagen tridimensional del entorno del vehículo.

- **Cámara**, (Bumblebee XB3**[7]**).

  Situada en la parte frontal del vehículo. Realiza grabaciones de vídeo de todo lo que se encuentre delante del vehículo para su posterior procesamiento en tiempo real.

- **IMU**

  Unidad de Medición Inercial. Mediante el uso de un acelerómetro junto a un giroscopio, y el adecuado procesamiento de las señales generadas por ambos dispositivos, se puede conocer la posición y velocidad del vehículo. Su principal utilidad es la de corregir errores en la señal GPS, que puede verse interrumpida si no dispone de la señal procedente de los satélites o a la perdida de cobertura de la señal GPRS por donde se reciben las correcciones diferenciales.

- **GPS**

  Sistema de posicionamiento global. Para este proyecto se ha utilizado un GPS diferencial con una estación base montada en el Campus externo de la Universidad de Alcalá, logrando así una mayor precisión.

- **Encoders**

  Sensores ubicados en las ruedas traseras que se encargan de medir el número de giros que han realizado las ruedas. Gracias a estos datos y a su correspondiente procesamiento se puede conocer, mediante un cálculo diferencial de las dos ruedas, la velocidad del vehículo y su posición sabiendo la relación de las ruedas y la distancia entre ambas. De la misma manera que con la IMU, se ha desarrollado un sistema que mediante un filtro de Kalman**[8]** corrija la posición del GPS, consiguiendo así una medida más precisa al funcionar con tres sistemas complementarios que proporcionan la posición robusta del vehículo en tiempo real.

Una vez que estos sensores han obtenido los datos, es el momento de procesarlos para conocer la posición del vehículo y los obstáculos situados en su entorno. Con todo ello y con un mapa de Lanelets (Fig. 1.1) (mapa construido específicamente para esta tarea, que describe la situación de los carriles transitables por el vehículo creando una "línea de obstáculos" con los otros carriles para que el coche no circule por ellos)**[9]**, es posible realizar una navegación simulada del vehículo para así después poder pasar al entorno real en el que entra en función el control a bajo nivel.

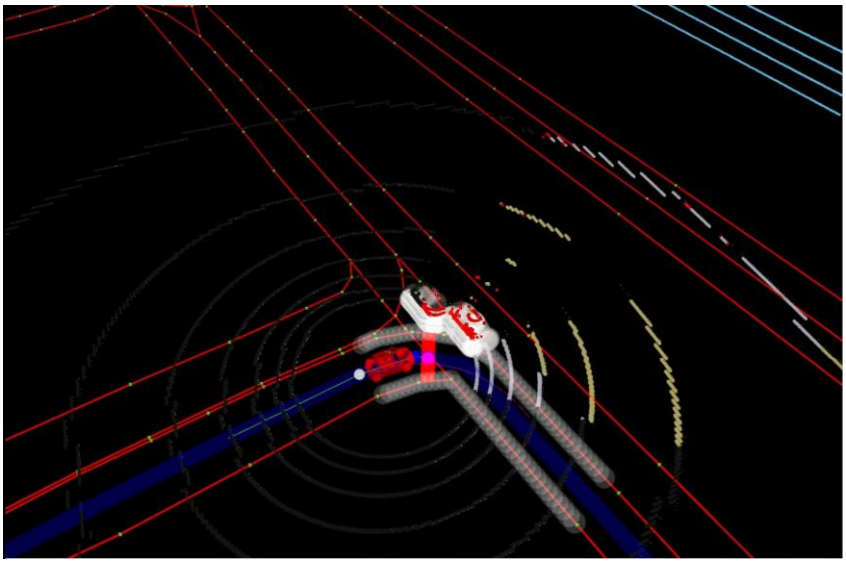


Fig. 1.1: Mapa de Lanelets

Para llevar a cabo maniobras de conducción es necesario introducir una capa de control a alto nivel que opera el vehículo. Para la gestión de todos los módulos de precepción, posicionamiento, control y navegación se utiliza ROS**[8]** (Sistema Operativo Robótico), un entorno de código abierto para el desarrollo de software de robots, en concreto, la versión utilizada en esta investigación es la Kinetic Kame.

En la Fig. 1.2 se muestra el vehículo utilizado en el proyecto con las modificaciones mecánicas necesarias para lograr la conducción autónoma, basado en el Tabby Evo**[11]**.

Fig. 1.2: SmartElderlyCar

1.3 Introducción al trabajo final de grado

Este trabajo se basa en adaptar un vehículo eléctrico para que pueda ser conducido de forma autónoma, proporcionándole la facilidad conocida como drive-by-wire.

Para ello se han desarrollado dos tareas principales:

- Diseño Hardware/Software de las unidades de control del acelerador y la dirección.
- Diseño de una unidad de conmutación entre ambos modos con un sistema de seguridad que permite pasar a modo manual en cuanto el conductor toque el volante, el freno o el acelerador.

En la Fig. 1.2 se muestra el diagrama de bloques de las fases asociadas a cada una de las tareas.

Fig. 1.3: Diagrama de bloques de las tareas

Capítulo 2: Control del acelerador y la dirección

2.1 Introducción al sistema implementado en el micro controlador

En este capítulo se detallarán los sistemas implementados y las modificaciones hardware necesarias para el control del acelerador y la dirección desde la unidad de control (ECU). Para ello se ha utilizado como elemento de control principal un micro controlador basado en Arduino, la Olimexino-STM32 (Fig. 2.1).

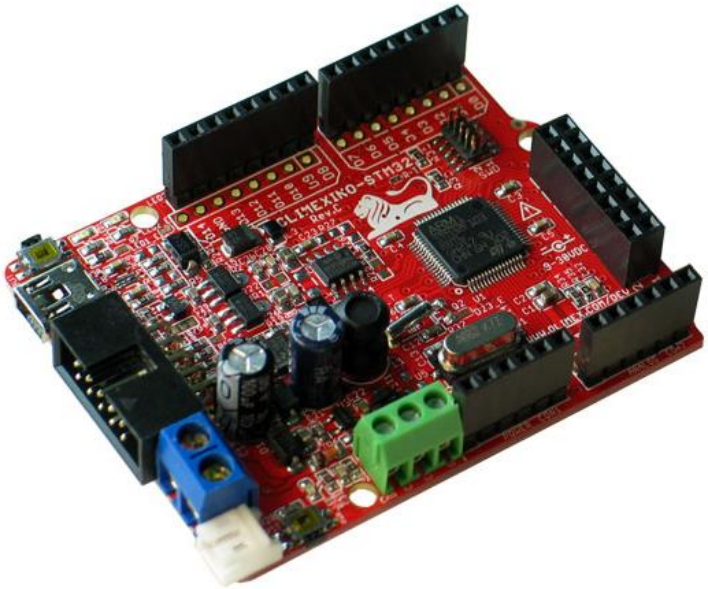


Fig. 2.1: Olimexino-STM32

Dispone de bus CAN, que es utilizado principalmente en la industria de la automoción y será un elemento final a tener en cuenta para la comunicación de todos los sistemas, adecuándose así a la industria actual. Además, incorpora la posibilidad de instalar un sistema operativo para conseguir controlar los diferentes periféricos en tiempo real sin que éstos se solapen.

El flujo de trabajo llevado a cabo ha seguido la siguiente estructura:

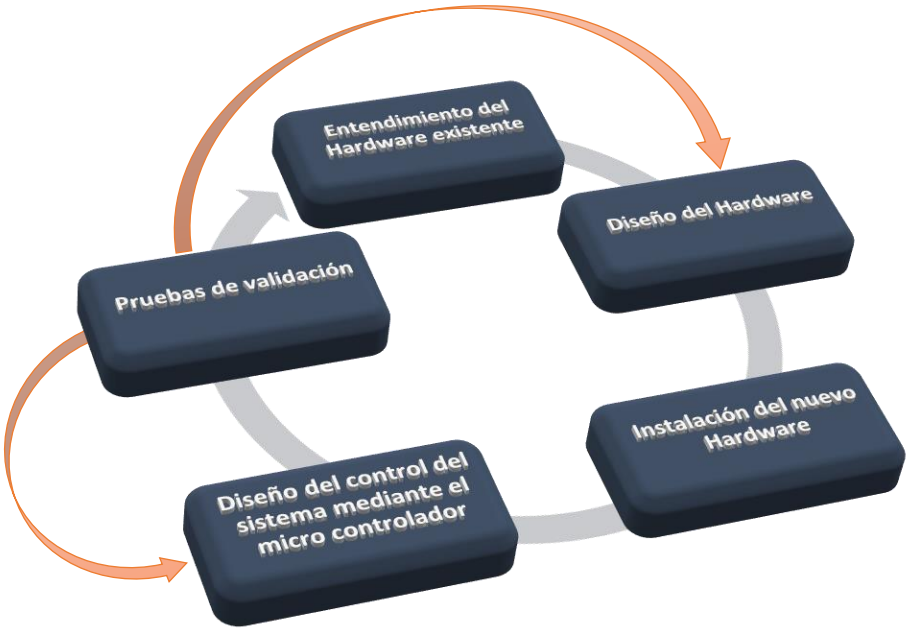


Fig. 2.2: Estructura de trabajo

- **Entendimiento del Hardware existente**: medida de las señales generadas por el vehículo y elementos que se pueden modificar sin ocasionar demasiados cambios en la estructura del vehículo.
- **Diseño del Hardware**: creación de circuitos impresos, cambios en la circuitería del vehículo, creación de sistemas de control, etc.
- **Instalación del nuevo Hardware**: recableado del vehículo, instalación de la nueva dirección y montaje del sensor de la dirección.
- **Diseño del control del sistema mediante el micro controlador**: creación del programa que dirija el control a bajo nivel, mediante la placa Olimexino-STM32.
- **Pruebas de validación**: realización de las diferentes pruebas para un funcionamiento óptimo. Este paso ocasionalmente puede requerir retroceder y rediseñar algunas partes cuando el resultado no sea el esperado.

2.2 Cómo está dirigido el micro controlador

Debido a que el micro controlador hace uso del mismo periférico para varias tareas y se necesita que todo el conjunto funcione lo más cercano posible al tiempo real, es necesario implementar un sistema operativo que gobierne todos los procesos que se han de ejecutar, evitando la concurrencia en el uso de periféricos y permitiendo que estas tareas se realicen lo más rápidamente posible. El sistema operativo utilizado es el FreeRTOS**[12]**.

Para el control de las tareas que hacen uso del I_2C se ha implementado un semáforo. Éste no permitirá el acceso al periférico a otra tarea mientras haya una ejecutándose, para así asegurarse de que solamente una función está empleando ese recurso.

Para la utilización del semáforo en esta aplicación, se le debe dar un nombre (Fig. 2.3) antes de crearlo, y comprobar que no se haya creado antes, para después habilitarlo, es decir, permitir que las tareas puedan acceder a él (Fig. 2.4).

```
// Declare a mutex Semaphore Handle which we will use to manage the I2C.
// It will be used to ensure only only one Task is accessing this resource at any time.
SemaphoreHandle_t xI2CSemaphore;
```

Fig. 2.3: Declaración del Semáforo

```
if ( xI2CSemaphore == NULL )  // Check to confirm that the I2C Semaphore has not already been created.
{
  xI2CSemaphore = xSemaphoreCreateMutex();  // Create a mutex semaphore we will use to manage the I2C
  if ( ( xI2CSemaphore ) != NULL )
    xSemaphoreGive( ( xI2CSemaphore ) );  // Make the I2C available for use, by "Giving" the Semaphore.
}
```

Fig. 2.4: Creación del Semáforo

Con el semáforo disponible, es momento de crear las tareas que han de ser gobernadas por éste (Fig. 2.6). Además, se definen las tareas que van a hacer uso del puerto serie, tanto para la recepción de instrucciones por parte del sistema de control como para la emisión de datos por el terminal, con el fin de poder ver ciertos valores medidos por los sensores y observar si se corresponden con los movimientos del vehículo, proporcionando una realimentación al sistema de control.

Todas las tareas han sido configuradas con la misma prioridad, por lo que ninguna se debe ejecutar por encima de la otra, esperando a que finalice una tarea para comenzar con la siguiente.

Para detallar el funcionamiento del semáforo se describirá la función encargada del control del acelerador (Fig. 2.5). En ella, antes de usar el periférico I_2C para transmitir el valor de la tensión que debe proporcionar el DAC al acelerador, se comprueba si el semáforo está habilitado, es decir, si ninguna otra tarea se encuentra en ejecución. Si la hubiera, esperaría 5 pulsos antes de volver a comprobar si se encuentra libre. Una vez esté libre transmitirá el valor al DAC y liberará el semáforo para que otra tarea pueda acceder al mismo.

```
    xTaskCreate(vREADSerialTask,
                "Task1",
                configMINIMAL_STACK_SIZE,
                NULL,
                tskIDLE_PRIORITY + 2,
                NULL);

    xTaskCreate(vANGLEControlTask,
                "Task2",
                configMINIMAL_STACK_SIZE,
                NULL,
                tskIDLE_PRIORITY + 2,
                NULL);

    xTaskCreate(vSPEEDControlTask,
                "Task4",
                configMINIMAL_STACK_SIZE,
                NULL,
                tskIDLE_PRIORITY + 2,
                NULL);

    xTaskCreate(vWRITESerialTask,
                "Task5",
                configMINIMAL_STACK_SIZE,
                NULL,
                tskIDLE_PRIORITY + 2,
                NULL);

    vTaskStartScheduler();
}
```

Fig. 2.5: Creación de las tareas

```
// See if we can obtain or "Take" the Serial Semaphore.
// If the semaphore is not available, wait 5 TickType_tticks of the Scheduler to see if it becomes free.
if ( xSemaphoreTake( xI2CSemaphore, (  ) 5 ) == pdTRUE )
{
  dac.setVoltage(value_dac, false);      //ERROR !!!!!!!!!!!!!!!!!!!!!!!!
  xSemaphoreGive( xI2CSemaphore ); // Now free or "Give" the I2C for others.
}
```

Fig. 2.6: DAC controlado por el semáforo

2.3 Recepción de datos por el puerto serie procedentes del sistema de control

Debido a que el vehículo debe ser dirigido por un sistema de sensores más complejos que los que determinen el movimiento de éste, es necesario establecer una comunicación entre ambos sistemas para optimizar el control fusionando el alto nivel con el bajo. Se recibe una cadena procedente del puerto serie cuyo formato tiene que ser (ángulo,velocidad_rpm\n). Si la cadena no es enviada de esta manera no se producirán los resultados esperados.

Además, los valores recibidos serán establecidos dentro de un cierto rango, de manera que no se reciban resultados irrealizables o que no sean convenientes para unas pruebas seguras de cara al movimiento del vehículo. Por ello, los ángulos recibidos estarán comprendidos entre -470 y 510 grados, y la velocidad entre 0 y 1.000 revoluciones por minuto. Con esta limitación no se podrá obtener una velocidad mayor de 18 km/h en conducción autónoma. En un futuro esta limitación será suprimida para así poder alcanzar velocidades mayores.

2.4 ¿Qué debe controlar el micro controlador?

2.4.1 Acelerador

Se han realizado ciertas mediciones en las señales generadas por el acelerador para comprobar cómo funciona el vehículo y poder generar las mismas señales para el control automático. El acelerador consta de cinco señales: el acelerador, su Inhibit, dos masas y la alimentación. Habría que generar estas mismas señales externamente.

El comportamiento de cada señal es el siguiente.

- Acelerador: varía entre 0,5 y 4,5 voltios.
- Inhibit: en reposo se encuentra a 12 voltios. Cuando el conductor actúa sobre el acelerador cambia a 0 voltios.

Todo ello alimentado a 12 voltios y con dos referencias de masa.

2.4.2 Dirección

Es necesario realizar el giro del volante de forma que las ruedas cambien de dirección. Para ello, se ha montado una servodirección de un vehículo comercial (Opel Corsa). De esta manera se podrá girar el volante mediante el uso de señales, ya que ésta consta de un motor que normalmente se encarga de la asistencia al giro, permitiendo al conductor aplicar menos fuerza de la necesaria para mover las ruedas. En este caso se encargará de efectuar el giro completo únicamente usando el motor sin asistencia del conductor.

Además del accionamiento del movimiento del volante, es necesario implementar un sensor de ángulo para poder determinar los grados que se debe girar el volante en cada movimiento, consiguiendo así el giro solicitado por el control de alto nivel.

Para poder realizar un proyecto OpenSource ha sido necesaria la fabricación de un sensor de ángulo, capaz de determinar los grados el ángulo que ha girado el volante para así poder tener una realimentación de su posición y realizar un control óptimo.

2.5 Modificaciones Hardware

Para la realización de las tareas descritas es necesario llevar a cabo una serie de modificaciones en el Hardware inicial que permitan controlar el vehículo.

2.5.1 Acelerador

Para la etapa del acelerador ha sido necesario cambiar el conexionado inicial del vehículo, ya que, al comienzo, las señales provenientes del acelerador se dirigían directamente al motor, por lo que para poder realizar el control de éste se cortaron los cuatro cables y se colocaron dos conectores, uno macho y el otro hembra (2.7). De esta manera, se podría establecer el manejo del acelerador de forma manual conectando ambos cables, o dirigir ambos al sistema de control para su correcto multiplexado, permitiendo conducir las señales tanto provenientes del acelerador como las generadas en el micro controlador, en función del modo de conducción en el que se encuentre el vehículo.

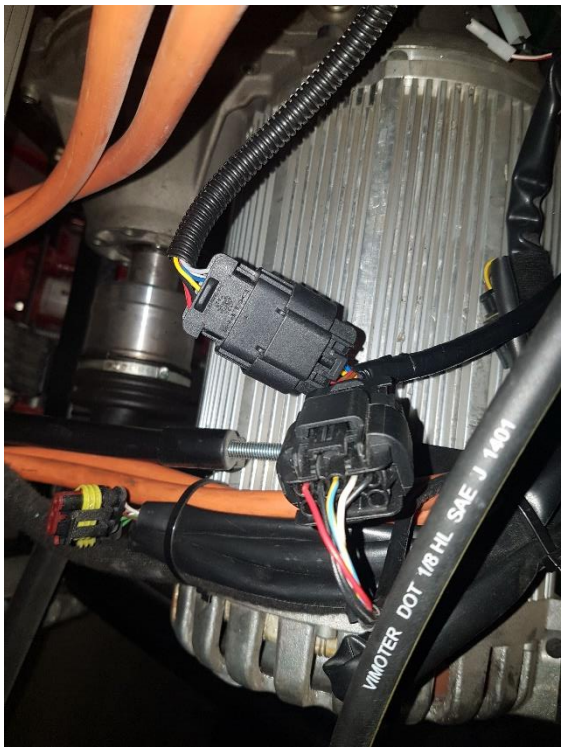


Fig. 2.7: Conectores acelerador

Además, para poder conocer la velocidad de giro del motor ha sido necesario realizar la misma tarea con el encoder incorporado en éste. Para no realizar un mal conexionado por error y evitar posibles fallos se han colocado conectores diferentes. (Fig. 2.8)

Fig. 2.8: Conector Encoders

2.5.2 Dirección

Para la etapa de la dirección ha sido necesario modificarla, cambiando la inicial por la dirección de un Opel Corsa, ya que, al disponer de servodirección, podía ser girado el volante mediante señales. (Fig. 2.9)

Fig. 2.9: Servodirección

De esta manera, mediante el uso de un Driver, el volante puede girar en ambos sentidos sin que el conductor haga uso de éste.

Además, se ha modificado esta dirección para acoplar un sensor de ángulo y así conocer la posición del volante, y por la tanto la de las ruedas en todo momento, realimentando al sistema y realizando un control óptimo. (Fig. 2.10)

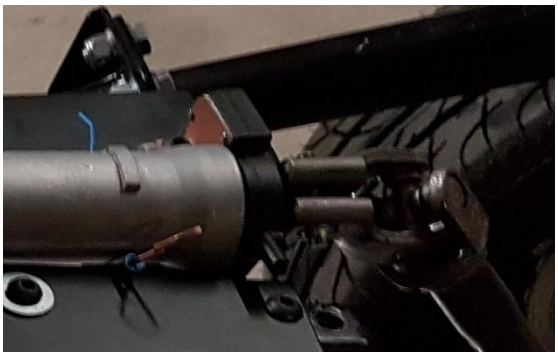


Fig. 2.10: Sensor de ángulo

2.6 Implementación

2.6.1 Acelerador

Debido a la imposibilidad del micro controlador de generar una señal de más de 3,3 voltios, y siendo necesario obtener una señal del acelerador entre 0,5 y 4,5 voltios, se ha implementado un circuito externo mediante un conversor digital analógico, capaz de suministrar el voltaje necesario (MCP4725-A0**[13]**).

Se ha utilizado este dispositivo debido a que dispone de unas librerías**[14]** concretas para trabajar con Arduino, además de ser capaz de suministrar la señal necesaria, ya que técnicamente tiene la posibilidad de generar una señal entre 0 y 5 voltios, por lo que se encuentra dentro la zona de trabajo.

Para su utilización es necesario añadir las librerías creadas para trabajar en Arduino:

#include <Adafruit_MCP4725.h>

Así se dispone tanto de las funciones necesarias para su configuración como de las de ejecución. Después se debe inicializar la clase *'dac'* que es utilizada para ejecutar las funciones dentro de la librería del MCP4725-A0 y crear las variables que se emplean tanto para la generación de las señales como para los cálculos de velocidades. (Fig. 2.11)

```
//--------------THROTTLE(DAC)-----------------------//
Adafruit_MCP4725 dac;
unsigned int value_dac = 400; // 0 (0V) <= value_dac <= 4095 (5V) -> Lineal speed (throttle) -> 400 (0.5V) <= value_dac <= 3800 (4.5V)
unsigned int target_rpm = 0, target_rpm_aux;
```

Fig. 2.11: Declaración del DAC

Una vez que las variables han sido creadas, es necesario realizar la configuración del DAC (Fig. 2.12). Acudiendo a la hoja de características del dispositivo se obtiene que para la configuración del I_2C el código del dispositivo es "1100" y, al ser el dispositivo utilizado el A0, se configura la dirección como 0. Por ello, se realiza "dac.begin(0x60);". De esta manera ya estará configurado el dispositivo. Se

establece el valor inicial en 400, que equivale a 0,5 voltios, simulando que el acelerador se encuentra en reposo.

```
//--------------THROTTLE(DAC+IC2)-----------------------//

// For Adafruit MCP4725A1 the address is 0x62 (default) or 0x63 (ADDR pin tied to VCC)
// For MCP4725A0 the address is 0x60 or 0x61  <---
// For MCP4725A2 the address is 0x64 or 0x65
dac.begin(0x60);
dac.setVoltage(value_dac, false);
```

Fig. 2.12: Inicialización del DAC

Debido a que ha sido configurado como A0, a la hora de montar el dispositivo (Fig. 2.13) se ha conectado con masa el pin 6 correspondiente a A0 mediante el uso de una resistencia para evitar problemas generados por las impedancias de entrada.

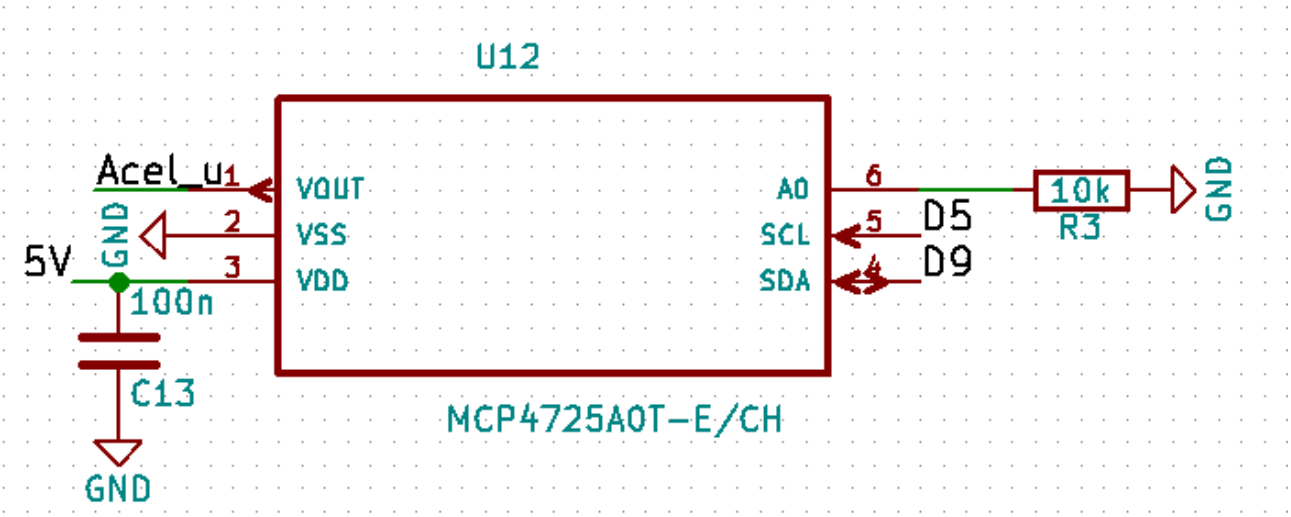


Fig. 2.13: Conexión del DAC

Con el dispositivo configurado, es momento de realizar los algoritmos que controlen el acelerador. Para ello se debe distinguir entre los datos leídos en el vehículo y los que se reciben desde el control a alto nivel para dirigir el vehículo mediante el puerto serie.

Se definen varias funciones para el control de esta unidad de control:

- **Control de velocidad** (Fig. 2.14)

  En esta función se realiza el envío de datos al DAC. Para ello, se necesitan los datos de velocidad de las ruedas y los recibidos por el puerto serie para la realización del controlador de esta unidad.

```
static void vSPEEDControlTask(void *pvParameters) {
  for (;;) {
    error_speed = target_rpm - wheelSpeed;

    if (error_speed > 0)
      value_dac = (error_speed * k_control_speed) + 600;  //De donde vienen esos 600
    else
      value_dac = 400; //frenamos con el motor

    //Lineal speed (throttle) -> 400 (0.5V) <= value_dac <= 3800 (4.5V)
    if (value_dac < 400) value_dac = 400;
    if (value_dac > 3800) value_dac = 3800;

    // See if we can obtain or "Take" the Serial Semaphore.
    // If the semaphore is not available, wait 5 ticks of the Scheduler to see if it becomes fr
    if ( xSemaphoreTake( xI2CSemaphore, ( TickType_t ) 5 ) == pdTRUE )
    {
      dac.setVoltage(value_dac, false);      //ERROR !!!!!!!!!!!!!!!!!!!!!!!!
      xSemaphoreGive( xI2CSemaphore ); // Now free or "Give" the I2C for others.
    }
    vTaskDelay(50);
  }
}
```

Fig. 2.14: Función de control de velocidad

Con ambos datos, se calcula el error entre la medida realizada y la que debemos conseguir. Si este error es mayor de 0, debemos acelerar, debido a que la velocidad del vehículo es menor que la que se ha recibido desde el módulo de control. Por ello, se ha añadido un controlador proporcional basado en la siguiente regla (Ecu. 2.1).

Las constantes han sido calculadas de manera experimental. Será necesario realizar un ajuste posteriormente. La constante es de valor 30 y es necesario añadir una consigna de 600 debido a que, sin ella, no aceleraría para errores pequeños.

$$\text{value\_dac} = (\text{errorspeed} * \text{Constante\_proporcional}) + 600$$

Ecu. 2.1: Control velocidad

Si el error fuera menor que 0, significaría que el vehículo se desplaza a mayor velocidad de la requerida, por lo que fijaríamos el valor del DAC a 400, que indica que no se está pisando el acelerador, por lo que el vehículo frenaría al actuar el freno del motor.

Seguidamente, se realiza la comprobación de que no se han sobrepasado los valores máximos de tensión en el DAC, estableciendo los límites de saturación en 400 y 3800, que son los valores para 0,5 y 4,5 voltios respectivamente.

Por último, esta función hace uso del I_2C de la placa, por lo que es necesario utilizar los semáforos que nos proporciona el sistema operativo para no tener problemas de concurrencia al ser utilizado el mismo periférico por varias funciones del programa.

- **Lectura de velocidad** (Fig. 2.16)

  La siguiente función trabaja como una interrupción, por lo que inicialmente habrá que configurarla (Fig. 2.15). De la manera que se ha inicializado, cada vez que se obtenga un pulso en el canal B correspondiente al encoder del motor se procederá a ejecutar la función de lectura de velocidad.

```
//--------------READ SPEED(MOTOR's ENCODER)-----------------------/
pinMode(ENCODER_CH_A, INPUT);
pinMode(ENCODER_CH_B, INPUT);
attachInterrupt(ENCODER_CH_B, readSpeed, RISING);
//-----------------------------------------------------//
```

Fig. 2.15: Entradas del encoder

Cada vez que salte la interrupción se realizará una medida del tiempo transcurrido desde que se encendió el sistema hasta que se produce la misma. Calculando la diferencia con la medida anterior, y dado que este valor es dado en micro segundos, se obtiene el tiempo transcurrido entre dos pulsos del encoder en micro segundos.

Con la medida del tiempo entre dos pulsos se realiza una conversión a grados por segundo, que son las unidades en las que se recibirán los datos del sistema de control a alto nivel. Para esta transformación se utiliza (Ecu. 2.2).

$$Wheel\_speed = \frac{1000000 * 360}{Tiempo\_entre\_pulsos * 6 * 64} = \frac{937500}{Tiempo\_entre\_pulsos} [º/s]$$

Ecu. 2.2: Cálculo de la velocidad de las ruedas

Esta conversión se realiza porque cada revolución en las ruedas equivale a 6 revoluciones del motor y a que el encoder del motor debe contar 64 pasos para cada vuelta. Además, se debe convertir de micro segundos a segundos.

De cara a las pruebas de test realizadas en el desarrollo del sistema, se realiza una conversión a kilómetros por hora (Ecu. 2.3). El diámetro de la rueda para el cálculo de la velocidad es 0,5735 m.

$$Speed_{kmh} = Wheel\_speed * (\frac{\Pi * D_{rueda}}{360}) * (\frac{3600}{1000}) = Wheel\_speed * (\frac{2}{111})[km/h]$$

Ecu. 2.3: Cálculo de la velocidad en km/h

```
void readSpeed() {
  new_time = micros();
  duration = new_time - old_time;
  old_time = new_time;
// frecuency[Hz] = 1000000/duration[
  wheelSpeed = 937500 / duration; //

  speed_kmh = wheelSpeed * 2 / 111;
}
```

Fig. 2.16: Cálculo de la velocidad

Una vez implementado, se ha medido los resultados obtenidos, comparándolos con la consigna enviada desde el sistema de control (Fig. 2.17). El resultado obtenido para el controlador utilizado es una respuesta con sobre impulso.

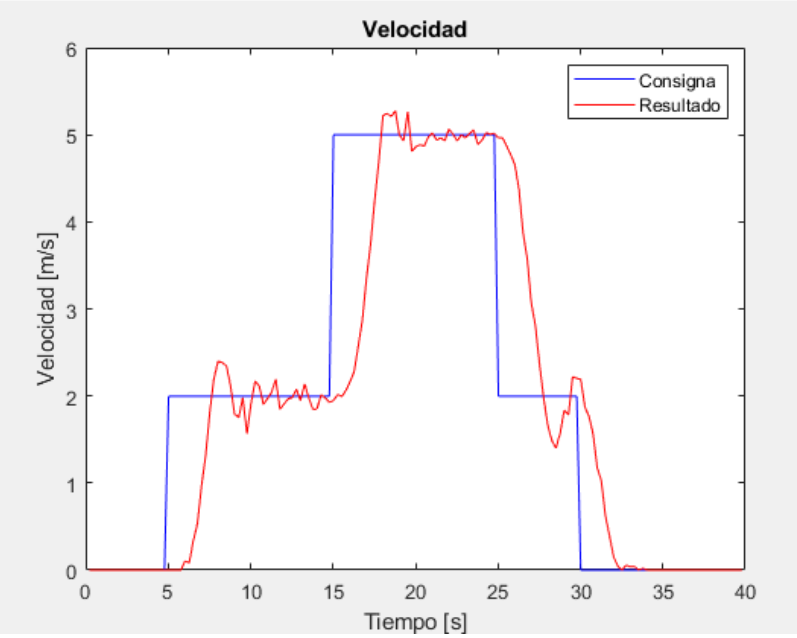


Fig. 2.17: Resultados acelerador

2.6.2 Dirección

Para su implementación se ha utilizado un Driver (IBT-2**[15]**) (Fig. 2.18) que es un puente en H compuesto de **dos semi-puentes**. De esta manera se podrá controlar el motor de corriente continua de la servodirección desde el micro controlador. Este Driver permite otorgar la corriente necesaria en los motores de corriente continua, siendo la alimentación de 12 voltios, que es la que se puede obtener del vehículo sin necesidad de modificaciones.

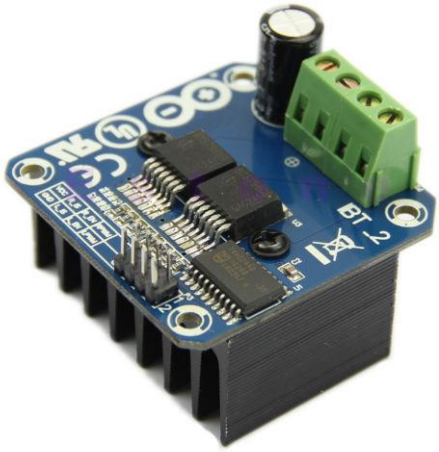


Fig. 2.18: IBT-2

Para la configuración del Driver se utilizan los siguientes pines del micro procesador (Fig. 2.19):

- D2: PWM en el sentido horario.
- D6: PWM en el sentido antihorario.
- D7: enable del Driver.

```
//--------------MOTOR'S DRIVER(SERVO->IBT_2)-----------------------//
#define Cw_PWM PA0   // PA0->D2; Clockwise connected to IBT-2 pin 1 (RPWM)
#define CCw_PWM PA8  // PA8->D6; counterclockwise connected to IBT-2 pin 2 (LPWM)
#define Inhibit_Servo PA9  // PA9->D7; inhibit servo
unsigned int PWM_value = 200; // 0 <= PWM_value <= 255 -> Rotational speed (SERVO)
```

Fig. 2.19: Definición de la servodirección

Para configurar el Driver (Fig. 2.20) se deben seleccionar los pines como las salidas PWM, que serán las encargadas de dirigir la velocidad del motor. Además, para habilitarlo se deben establecer como valor a nivel alto, es decir 3,3 voltios, las dos entradas de enable del Driver.

```
//--------------MOTOR'S DRIVER-------------------------//

pinMode(Cw_PWM, OUTPUT);
pinMode(CCw_PWM, OUTPUT);
pinMode(Inhibit_Servo, OUTPUT);
digitalWrite(Inhibit_Servo, HIGH);
```

Fig. 2.20: Configuración de la Servodirección

Seguidamente se definen las funciones que gestionarán el proceso del control de la dirección.

- **Medida de ángulo** (Fig. 2.26)

  Para la lectura del ángulo se debe especificar inicialmente el sistema realizado. Para ello, se ha construido un sensor**[16]** capaz de medir el ángulo del volante.

  Sensor de ángulo

  Para la creación de este sensor se ha implementado un **sistema de ruedas con un número diferente de dientes** (Fig. 2.21) y se ha acoplado un sensor de ángulo (KMZ41**[17]**) a las ruedas de menor tamaño, por lo que, mediante unas relaciones matemáticas, es posible conocer el ángulo desplazado por la dirección.

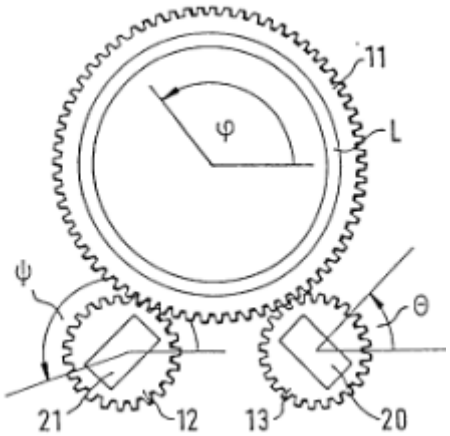


Fig. 2.21: Sensor de ángulo

El sensor (KMZ41) proporciona dos señales dependientes del ángulo en el que se encuentre (Ecu. 2.4 y 2.5) (Fig. 2.22).

Vo1 = V_peak * cos(2α) + Voff1	Vo2 = V_peak * sin(2α) + Voff2
Ecu. 2.4: Tensión de salida 1	**Ecu. 2.5:** Tensión de salida 2

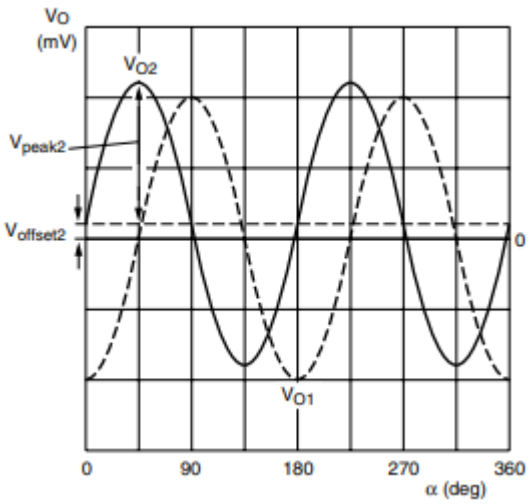


Fig. 2.22: Salida de los sensores de ángulo

Para medir estas señales se utiliza un conversor analógico digital (ADS1115**[18]**), leyendo en modo diferencial para así obtener el valor real en cada momento. Para poder trabajar con ellos se deben añadir sus librerías**[19]**. Además, es necesario hacer una corrección de offset, que se ha realizado probando el sistema, hasta calibrarlo. Ha sido necesario realizar cuatro calibraciones debido a que cada lectura contenía un offset diferente.

Para la medida de ambos sensores se ha generado una placa de circuito impreso con todos los elementos necesarios, con un reducido tamaño para que puedan ubicarse en la caja del sensor. Tendrá dos sensores KMZ41, uno para cada engranaje, y dos ADS1115 para medir los valores producidos por los sensores. Como se necesita que ambos ADC se comuniquen con el micro controlador mediante I_2C, se debe configurar la dirección de cada uno para que no se solapen. Por ello, en uno se llevará la entrada ADDR a masa mediante una resistencia y en el otro se llevará a 3.3 voltios mediante una resistencia. Así se consigue que sus direcciones

sean 1001000 y 1001001. Con estos datos se deben configurar la dirección de ambos en el código para que corresponda con lo establecido físicamente.

Una vez obtenidos ambos valores de cada sensor, se debe calcular el ángulo que ha girado cada uno, no sin antes pasar esta señal por un filtro RC. Es en este momento cuando se aprovecha la medida diferencial del sistema ya que, al no conocer exactamente el valor de pico de la tensión medida (debido a que esta depende de la temperatura del entorno), debemos conseguir una medida en la que ésta no influya (Ecu. 2.6).

$$Ángulo = 0.5 * atan\frac{Vo2}{Vo1} = 0.5 * \text{atan}\frac{\text{Vpeak} * \sin(2\alpha)}{\text{Vpeak} * \cos(2\alpha)}$$

Ecu. 2.6: Cálculo del ángulo

Al realizar esta transformación, no afecta el valor de pico de la tensión, el cual depende de la temperatura. Esta medida se realizará en ambas ruedas y se realizará una conversión idéntica para evitar el problema de las variaciones debidas a la temperatura.

Para el cálculo del ángulo con el arco tangente se ha utilizado una función no incluida en los paquetes de Arduino (**Función atan2**), debido a la lentitud de ésta y a la necesidad de realizar estos cálculos lo más rápidamente posible para no añadir esperas al diseño que puedan perjudicar la funcionalidad del sistema en su conjunto. El cálculo del ángulo se debe realizar en dos tareas:

1. Debido a que la función por aproximaciones para el arco tangente se realiza entre 0 y 45 grados, se debe dividir la circunferencia en 8 cuadrantes. (Fig. 2.23). Para realizar esta aproximación se crea un algoritmo que se encarga de establecer el cuadrante en el que se encuentra el ángulo, utilizando los valores de las coordenadas (x e y) obtenidos de la lectura en modo diferencial de los ADC.

   Se establecen tres condiciones:

   - Si la coordenada x es negativa, se suma 4.
   - Si la coordenada y es negativa, se suma 2.
   - Si x es menor que y, se suma 1.

Con estas tres condiciones se puede situar el área de trabajo en cada uno de los ocho cuadrantes disponibles.

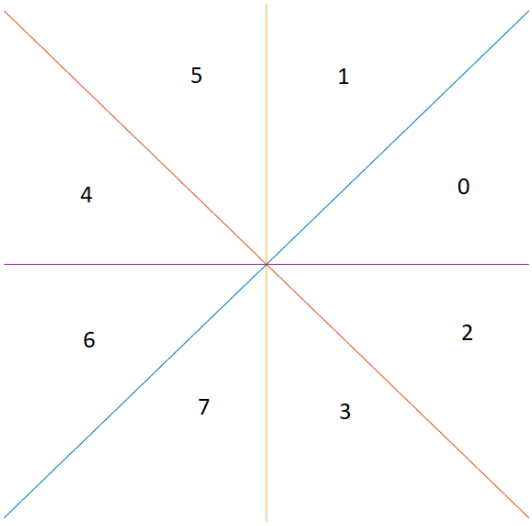


Fig. 2.23: Cuadrantes de la función atan2

Las operaciones llevadas a cabo según el cuadrante, con los valores absolutos de la coordenada (x e y) son:

1. $Atan(y, x)$
2. $90 - atan(x, y)$
3. $360 - atan(y, x)$
4. $270 + atan(x, y)$
5. $180 - atan(y, x)$
6. $90 + atan(x, y)$
7. $180 + atan(y, x)$
8. $270 - atan(x, y)$

De esta manera, intercambiando los valores de x e y, y añadiendo un offset, es posible situarse en todas las opciones de cuadrantes, realizando operaciones con ángulos entre 0 y 45 grados.

2. Función $atan(s, c)$ entre 0 y 45 grados.

   Operación del arco tangente entre 0 y 45 grados con precisión de un grado mediante el uso de un array para así conseguir un menor tiempo de ejecución a cambio de tener que guardar en memoria un array de 255 valores.

Para realizarlo sin utilizar la función proporcionada se debe trabajar con enteros, desplazar *s* ocho posiciones a la izquierda, que equivale a multiplicar por 2^8, para después dividir este valor por *c.*

El resultado indica la posición en el array (Fig. 2.25) del valor en grados buscado. Pero antes se debe saturar este valor en 255, que es el final del array, porque en los valores próximos a 45 grados el cálculo de la posición del array se desborda. Con esto se consigue un error menor de un grado (Fig. 2.24), por lo que el objetivo buscado, que era una precisión de grado, se habrá alcanzado.

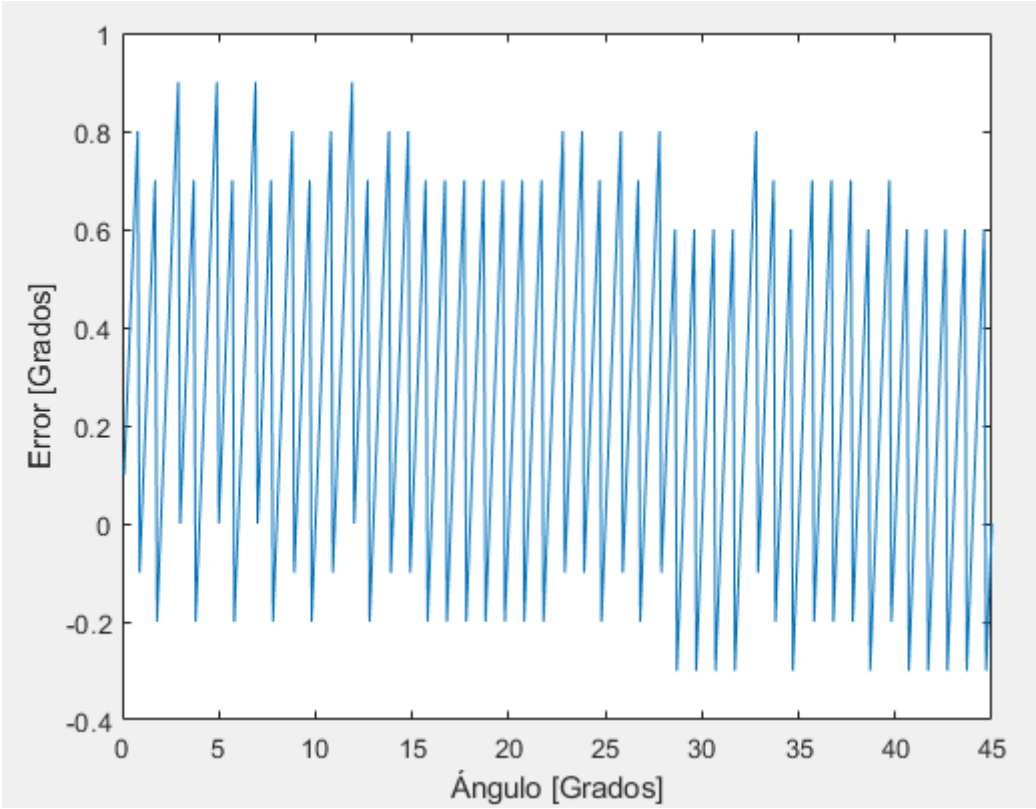


Fig. 2.24: Error de la función atan2

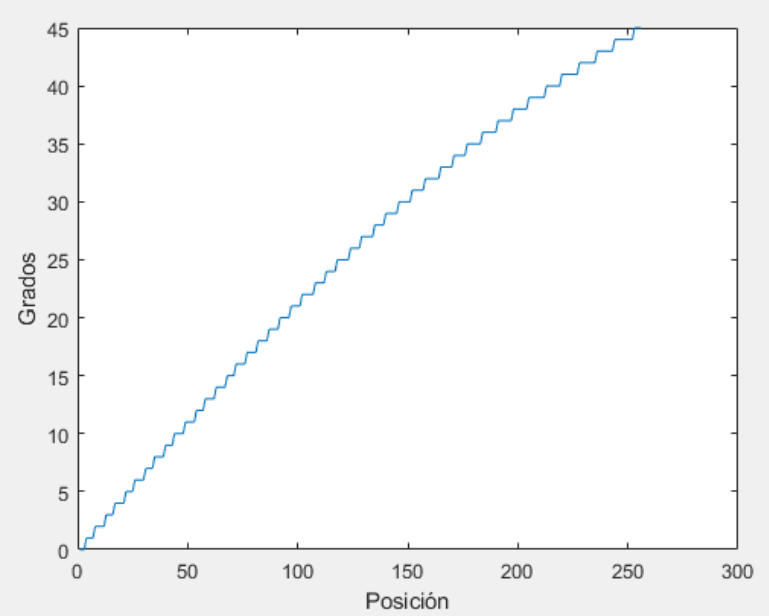


Fig. 2.25: Array de la función

Con ambos ángulos calculados, es momento de procesarlos para saber con precisión de grado el ángulo que ha girado el volante.

Se realiza la diferencia de los ángulos medidos en cada rueda. Si esta diferencia es menor que 0, se suma 180 grados, ya que implica que se ha dado una vuelta máxima debido a que los sensores solo miden hasta 180 grados. Tanto si se suman 180 grados como si no, se multiplica la diferencia de ángulos por una constante dependiente del número de dientes que tiene cada rueda (Ecu. 2.7).

$$Ángulo = \frac{m * \Psi + (m+2) * \theta}{2 * n} = (\Psi - \theta) * \frac{26}{3}$$

Ecu. 2.7: Cálculo del ángulo de la dirección

- m = 26, número de dientes de la rueda más pequeña.
- n = 42, número de dientes de la rueda principal.
- Ψ, giro de la rueda de menos dientes.
- θ, giro de la rueda con m+2 dientes.

```
void Calculate_Angle(void)
{
  //--------------ANGLE_SENSOR-----------------------//
  ads1_volt_0_1 = -7 + ads1.readADC_Differential_0_1();    //OFFSET=-7  <- ads1_volt_0_1 = -OFFSET+ads1.readADC_Differer
  ads2_volt_0_1 = -222 + ads2.readADC_Differential_0_1();  //OFFSET=-222
  ads1_volt_2_3 = -170 + ads1.readADC_Differential_2_3();  //OFFSET=-170
  ads2_volt_2_3 = -131 + ads2.readADC_Differential_2_3();  //OFFSET=-131
  //-------------------------------------//
  angle1 = atan2C(ads1_volt_2_3, ads1_volt_0_1) / 2;
  angle2 = atan2C(ads2_volt_2_3, ads2_volt_0_1) / 2;

  subtr = angle1 - angle2;
  if (subtr < 0)
    subtr = subtr + omega;

  angle = subtr * 26 / 3;
}
```

Fig. 2.26: Función de cálculo de ángulo

- **Control de ángulo** (Fig. 2.27)

  Función encargada del control de la dirección, calculando el ángulo en el que se encuentra el volante, la diferencia con el valor recibido del control a alto nivel y el control proporcional que dirige la PWM. De esta manera se elige qué función se debe realizar para el giro del volante.

```
static void vANGLEControlTask(void *pvParameters) {
  for (;;) {
    // See if we can obtain or "Take" the I2C Semaphore.
    // If the semaphore is not available, wait 5 ticks of the Scheduler to see if it becomes free.
    if ( xSemaphoreTake( xI2CSemaphore, ( TickType_t ) 5 ) == pdTRUE )
    {
      Calculate_Angle();
      xSemaphoreGive( xI2CSemaphore ); // Now free or "Give" the I2C for others.
    }

    error_ang = target_ang - angle;
    PWM_value = abs (error_ang) * k_control_ang + 20;
    if (PWM_value > 255)
      PWM_value = 255;

    if (error_ang > 10) {
      Go_CCW();
    }
    else if (error_ang < -10) {
      Go_CW();
    }
    else
      Stop();

    vTaskDelay(40);  //10ms read angle + 40ms dalay = 50ms
  }
}
```

Fig. 2.27: Función de control de la dirección

- **Parada** (Fig. 2.28)

  Ambas salidas se establecen a 0.

```
void Stop()
{
  analogWrite(Cw_PWM, 0);
  analogWrite(CCw_PWM, 0);
}
```

Fig. 2.28: Parada del volante

- **Giro en sentido horario** (Fig. 2.29)

  La salida de giro en sentido antihorario se establece a 0, mientras que la de sentido horario se fija con el valor calculado, siempre añadiendo un retardo entre ambas acciones por seguridad para evitar fallos en caso de que ambos puentes estén intentando girar el motor en sentidos opuestos.

```
void Go_CW()
{
  analogWrite(CCw_PWM, 0);
  vTaskDelay(5);
  analogWrite(Cw_PWM, PWM_value);
}
```

Fig. 2.29: Giro en el sentido horario

- **Giro en sentido antihorario** (Fig. 2.30)

  La salida de giro en sentido horario se establece a 0, mientras que la de sentido antihorario se fija con el valor calculado, siempre añadiendo un retardo entre ambas acciones por seguridad para evitar fallos en caso de que ambos puentes estén intentando girar el motor en sentidos opuestos.

```
void Go_CCW()
{
  analogWrite(Cw_PWM, 0);
  vTaskDelay(5);
  analogWrite(CCw_PWM, PWM_value);
}
```

Fig. 2.30: Giro en el sentido antihorario

Una vez implementado, se ha medido los resultados obtenidos, comparándolos con la consigna enviada desde el sistema de control (Fig. 2.31). En la respuesta se puede observar los tiempos de retardo generados por la dinámica del volante, ya que la consigna cambia en un instante, pero el volante debe girar y para ello necesita tiempo.

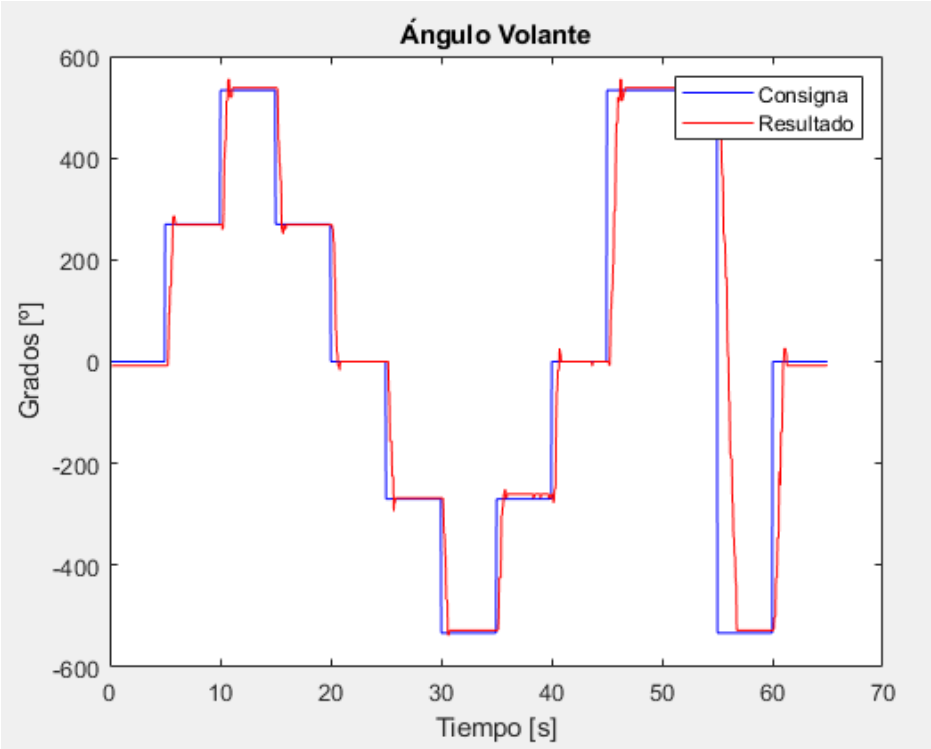


Fig. 2.31: Resultados dirección

Capítulo 3: Diseño de la placa impresa encargada de la conmutación del modo de conducción

3.1 Introducción a la placa de conmutación del modo de conducción

Sistema implementado para la conmutación del modo de conducción del coche entre manual y automático, actuando de multiplexor entre las señales generadas de forma nativa por el vehículo y las generadas vía software para el control automático.

Este sistema, además de ser indispensable para poder operar en ambos modos de conducción (automático y manual), es necesario principalmente por una cuestión de seguridad, dado que la legislación actual obliga al conductor a estar atento y preparado para coger los mandos del vehículo en cualquier momento. Por este motivo, el sistema diseñado otorgará los mandos del vehículo al conductor en cualquier momento que él crea conveniente. Para ello solo deberá actuar sobre el acelerador, freno o volante, y el sistema cambiará a modo de conducción manual automáticamente. De esta manera, siempre que el conductor detecte una situación inusual en la que el vehículo esté siendo dirigido automáticamente, pero no de forma correcta, éste podrá coger los mandos.

Además, el sistema deberá depender solamente de lógica, sin pasar por el micro controlador, el cual está más sujeto a posibles fallos, como reinicios, retardos del sistema, etc.

Este capítulo del TFG será comentado como un tutorial para el diseño de PCBs debido a que, a pesar de ser fundamental para un Ingeniero Electrónico, constituye una carencia del Grado en Ingeniería Electrónica y Automática Industrial.

3.2 Condiciones de diseño

3.2.1 Condiciones generales

El sistema tiene que funcionar como un elemento de seguridad, permitiendo siempre que el vehículo se encuentre en modo automático poder pasar al modo manual si así lo decide el conductor. Para implementar esta parte del sistema es necesario utilizar los Inhibits generados por el acelerador, freno y volante, de manera que cuando se active cualquiera de ellos se establezca el modo manual independientemente del modo en el que esté, y que éste se quede fijo hasta que el usuario vuelva a activar el modo automático mediante el pulsador.

Los Inhibit se activan a nivel bajo. Esto significa que cuando no se está interviniendo sobre el actuador se obtiene una salida de 12 voltios, mientras que cuando se activa se obtiene una salida de 0 voltios. Para la realización de esta función se ha creado un sistema pull-up en el vehículo, tal como se observa en la Fig. 3.1.

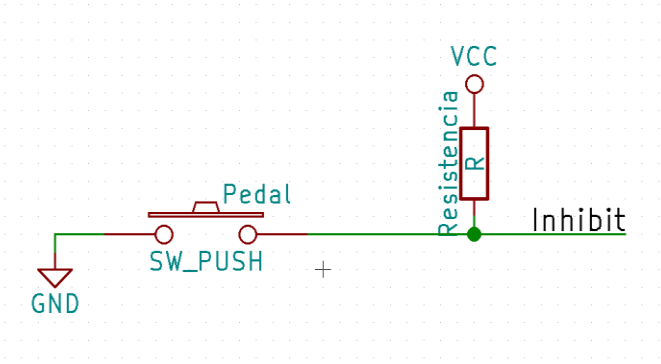


Fig. 3.1: Funcionamiento de los Inhibits

La tabla de verdad que dirigirá esta implementación se muestra a continuación (Tab. 3.1).

Inhibit_freno	Inhibit_acelerador	Inhibit_volante	Pulsador	Modo
0	0	0	0	Manual
0	0	0	1	Manual
0	0	1	0	Manual
0	0	1	1	Manual
0	1	0	0	Manual
0	1	0	1	Manual
0	1	1	0	Manual
0	1	1	1	Manual
1	0	0	0	Manual
1	0	0	1	Manual
1	0	1	0	Manual
1	0	1	1	Manual
1	1	0	0	Manual
1	1	0	1	Manual
1	1	1	0	Manual
1	1	1	1	Automático

Tab. 3.1: Modo de estado de conducción

Como se puede observar en la tabla, la única forma de que el sistema no se encuentre en modo manual es que el pulsador se haya activado y ninguno de los Inhibit esté activo.

El pulsador es activo a nivel alto, y ha sido necesario implementar un sistema para evitar falsos positivos debidos a rebotes proporcionados por el pulsador. Para la realización de esta función se ha utilizado una red RC (Fig. 3.2) que realizará una carga del condensador cuando el pulsador se active. A partir de ahí, será éste el encargado de fijar la tensión a la salida para evitar rebotes de la señal.

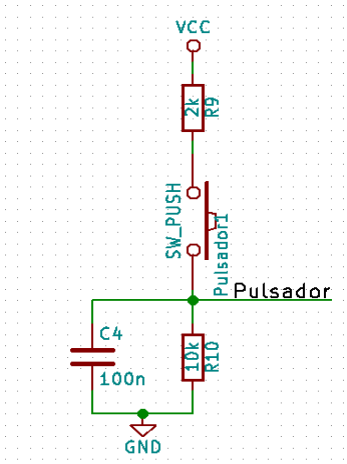


Fig. 3.2: Pulsador del cambio de modo

La forma de determinar si el sistema se encuentra en modo manual o automático es mediante un biestable JK, con la siguiente tabla de verdad (Tab. 3.2).

J	K	Q
0	0	Sin cambio
0	1	0
1	0	1
1	1	Cambio

Tab. 3.2: Biestable JK

Para el correcto funcionamiento del biestable, se conectará la entrada J con el bloque del pulsador (Fig. 26) y la entrada K con el bloque de los Inhibit (Fig. 3.4), de manera que cuando el pulsador se active establezca una salida a nivel alto (12 voltios), dejándola fija hasta que se intervenga sobre cualquiera de los tres actuadores. Para la realización del biestable JK se ha utilizado un bloque de puertas NAND (CD4093BPW**[20]**) que logra simplificar el número de componentes diferentes. Se ha seguido el esquema de la Fig. 3.3.

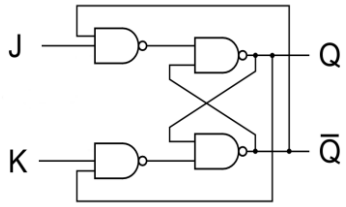


Fig. 3.3: Biestable JK con puestas NAND

La puerta CD4093BPW es un dispositivo CMOS de cuatro puertas NAND Trigger Schmitt**[21]** con dos entradas cada una de ellas. Es necesario que las puertas sean CMOS, debido a la necesidad de tener entradas y salidas en torno a los 12 voltios, por lo que se emplean estos dispositivos que pueden ser alimentados hasta 20 voltios. Que sean Trigger Schmitt es un añadido, utilizando la histéresis para prevenir el ruido que podría tapar a la señal original y que causaría falsos cambios de estado si los niveles de referencia y entrada fueran parecidos.

Para el correcto funcionamiento del bloque de los Inhibit (Tab. 3.3) es necesario implementar el siguiente circuito (Fig. 3.4).

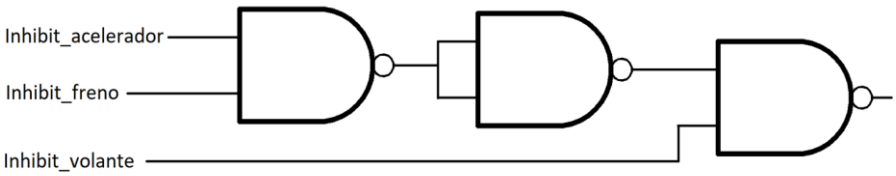


Fig. 3.4: Conexión de los Inhibits

Inhibit_acelerador	Inhibit_freno	Inhibit_volante	Salida_k
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Tab. 3.3: Comportamiento de la entrada K del biestable

De este modo, al estar por defecto los Inhibit a nivel alto (12 voltios), el conductor no accionará ninguno de los tres actuadores. La salida se deberá activar para pasar al modo manual en el momento en cualquiera de los tres cambie a nivel bajo, produciendo una conmutación en el biestable.

3.2.2 Multiplexores

Una vez que dispongamos de una señal que nos indique en qué modo se encuentra el sistema, (automático o manual), será necesario un sistema que seleccione entre las señales generadas por el vehículo y las generadas por el micro controlador.

Para ello se ha utilizado el dispositivo DG419DY**[22]**, que es un switch analógico con tecnología CMOS, por lo que permite funcionar con los voltajes necesarios hasta 12 voltios. Este es el motivo de la elección de este componente, además de que la resistencia de encendido no es elevada (20 ohmios).

El funcionamiento de este dispositivo es el siguiente. Dispone normalmente de dos entradas y una salida, y según como esté el pin de control del dispositivo, cortocircuitará la salida con una de las dos entradas. Esta puerta es bidireccional, por lo que, además de poder usarse como multiplexor, se puede emplear como demultiplexor, consiguiendo que una entrada pueda ser llevada a una de las dos salidas posibles, todo ello dirigido por el pin de control.

Para que funcione de la manera deseada será necesario realizar un conexionado concreto (Fig. 3.5). Se podrá elegir entre el pin 2 y el 8. Para este diseño el pin 8 se debe activar a nivel alto y el 2 a nivel bajo, por lo que el conexionado se ha efectuado de la siguiente manera:

1. Salida
2. Entrada nivel bajo
3. GND
4. 12 voltios
5. 12 ó 5 voltios, dependiendo de si la señal de control la proporciona algún elemento del PCB o el micro controlador
6. Señal de control del switch
7. GND
8. Entrada a nivel alto

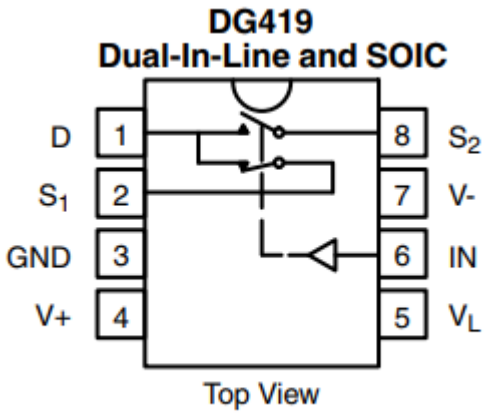


Fig. 3.5: DG419DY

De esta manera, hemos medido experimentalmente que el cambio de bajo nivel a alto se produce a los 4 voltios, por lo que para los Inhibit generados por la placa será necesario realizar una modificación consistente en alimentar V_L a 5 voltios. Así conseguimos que el nivel alto cambie a los 2,4 voltios y la placa basada en Arduino pueda provocar cambios de estado, ya que en las salidas solo puede establecer un valor de 3.3 voltios como máximo.

3.2.3 Condensadores de desacoplo**[23]**

Un condensador de desacoplo -también llamado condensador de derivación- se utiliza para desacoplar las señales de corriente alterna de una señal de corriente continua. Debido a que las tomas de alimentación contienen mucho ruido, es necesario colocar un condensador de la manera que se detalla en la Fig. 3.6.

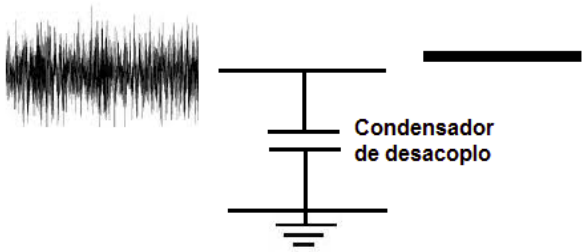


Fig. 3.6: Funcionamiento de los condensadores de desacoplo

La utilización de condensadores de desacoplo es especialmente importante en diseños como el que se ha realizado, debido a que en todo circuito lógico es necesario filtrar la alimentación, ya que ese ruido podría afectar a la manera en la que el circuito entiende una pulsación a nivel alto o bajo, provocando que el diseño no funcione de la manera requerida.

La forma en la que funcionan los condensadores es desviando la corriente alterna a tierra. Esto es debido a que otorgan menor resistencia cuanto mayor es la frecuencia de la señal, por lo que es perfecto para dejar pasar todo el ruido procedente de la alimentación y mantener en la alimentación la tensión de continua que es la ideal para el funcionamiento correcto de los componentes.

El condensador de desacoplo más común tiene un valor de 100 nF y es de tipo cerámico.

La forma de colocación óptima es lo más cerca posible del integrado para así reducir la inductancia generada por la pista y que no añada ruido.

3.3 Diseño del sistema

Una vez establecidas las condiciones, se llevará a cabo el diseño del sistema. Debido a que su finalidad es ser una placa de circuito impreso, se empleará un programa para el diseño de este tipo de placas.

El software utilizado ha sido Kicad, que es un software de código libre. Dispone de una serie de aplicaciones en las que se llevan a cabo las diferentes tareas para la realización y posterior fabricación de una placa de circuito impreso.

El esquema seguido (Fig. 3.7) para su diseño se muestra en el siguiente organigrama:

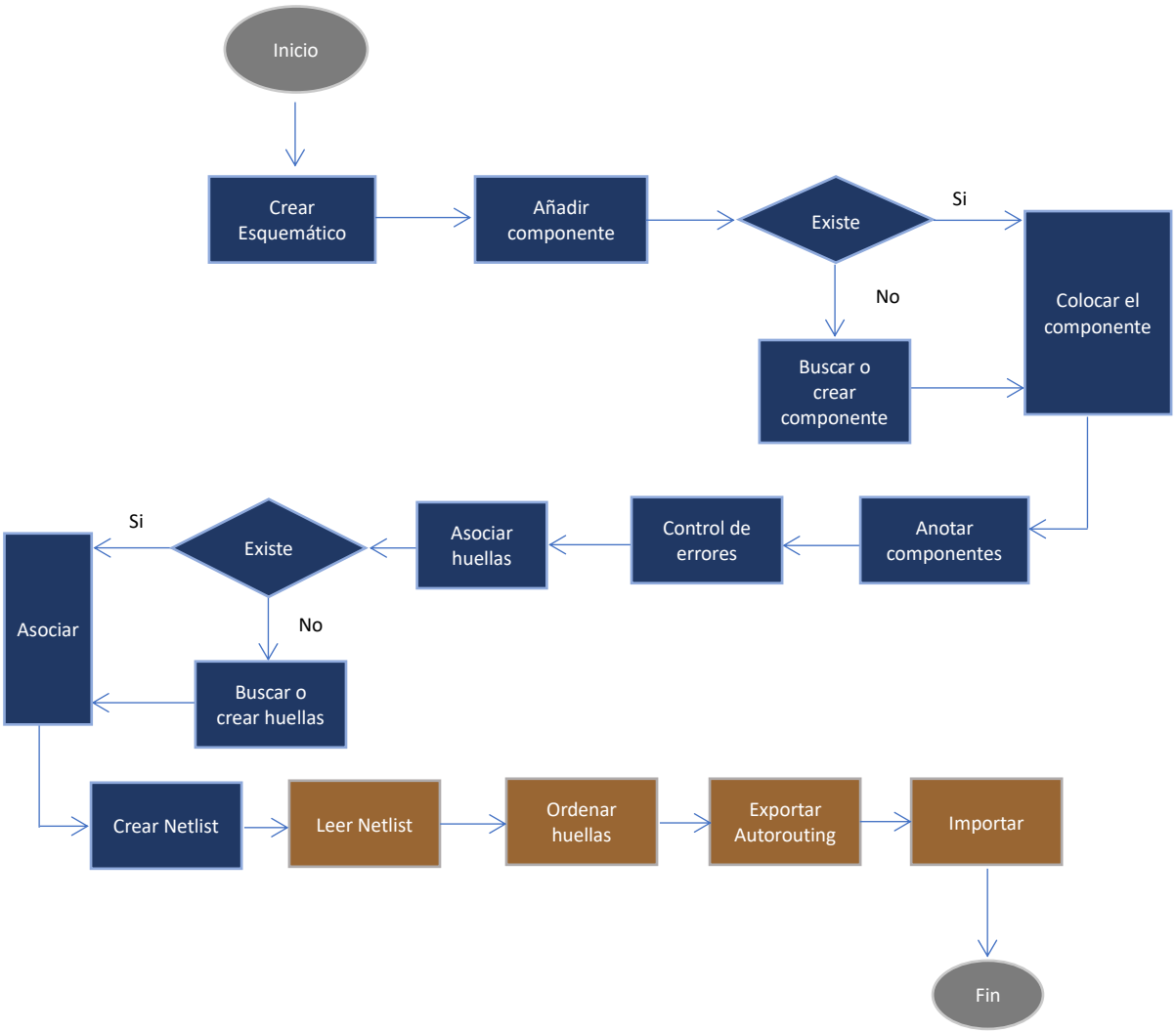


Fig. 3.7: Organigrama para el diseño de PCBs

3.3.1 Diseño del esquemático en Kicad (EESchema)

Primero se ha de comenzar por la realización del esquemático, ya que es el primer paso en la fabricación del sistema. Dentro de la aplicación de EESchema de Kicad se encontrará una plantilla en blanco en la que se tendrán que ir situando los diferentes componentes utilizados.

Pasos a realizar para cada componente:

1) Verificar si el componente existe en las librerías añadidas al proyecto.
   A. Si el componente existe, situarlo en el esquemático y continuar con el siguiente.
   B. Si el componente no existe, es necesario buscar una librería que disponga de éste.
      a) Se realiza una búsqueda del componente en https://www.snapeda.com/.
         - Si se encuentra el componente buscado, se descarga tanto la librería que contiene el componente en cuestión como su huella, que será utilizada en pasos posteriores del diseño.
         - De no encontrarlo, habrá que crear una librería que contenga este componente y su huella. (Este paso se detallará más adelante cuando se aborde la creación del conector de 2x16 pines).
      b) Una vez se dispone de la librería y la huella, es necesario añadirlas al proyecto. Para ello, se accede a Component Libraries y se añade a la carpeta donde se están guardando todos los archivos .lib necesarios por el diseño. Con esto ya se dispone de los componentes para poder ser añadidos al esquemático. Las librerías añadidas al proyecto son:
         - CD4093BPW
         - DG419DY
         - ADS1115IDGSR
         - MCP4725A0T-E_CH
         - Arduino-kicad-library-master
   C. Cuando ya se dispone de las librerías de todos los componentes, bien porque se hayan creado, porque se hayan descargado de repositorios de internet o porque ya estuviesen precargadas, se procederá a colocar todos los componentes en el esquemático.

2) Se comenzará montando la etapa para generar la señal K del biestable, es decir, la encargada del control de los Inhibits (Fig. 3.4). Para ello, se necesita el CD4093BPW, un condensador de 100 nF para desacoplar la alimentación, dos referencias de masa y una de 12 voltios, además de las señales de los tres Inhibit y cables para la unión de todo. El resultado es el de la Fig. 3.8. Como añadido necesario en Kicad, en las conexiones del integrado que no son usadas es necesario añadir que no tienen conexión para que no dé error a la hora de comprobar si hay algún fallo en la conexión del diseño.

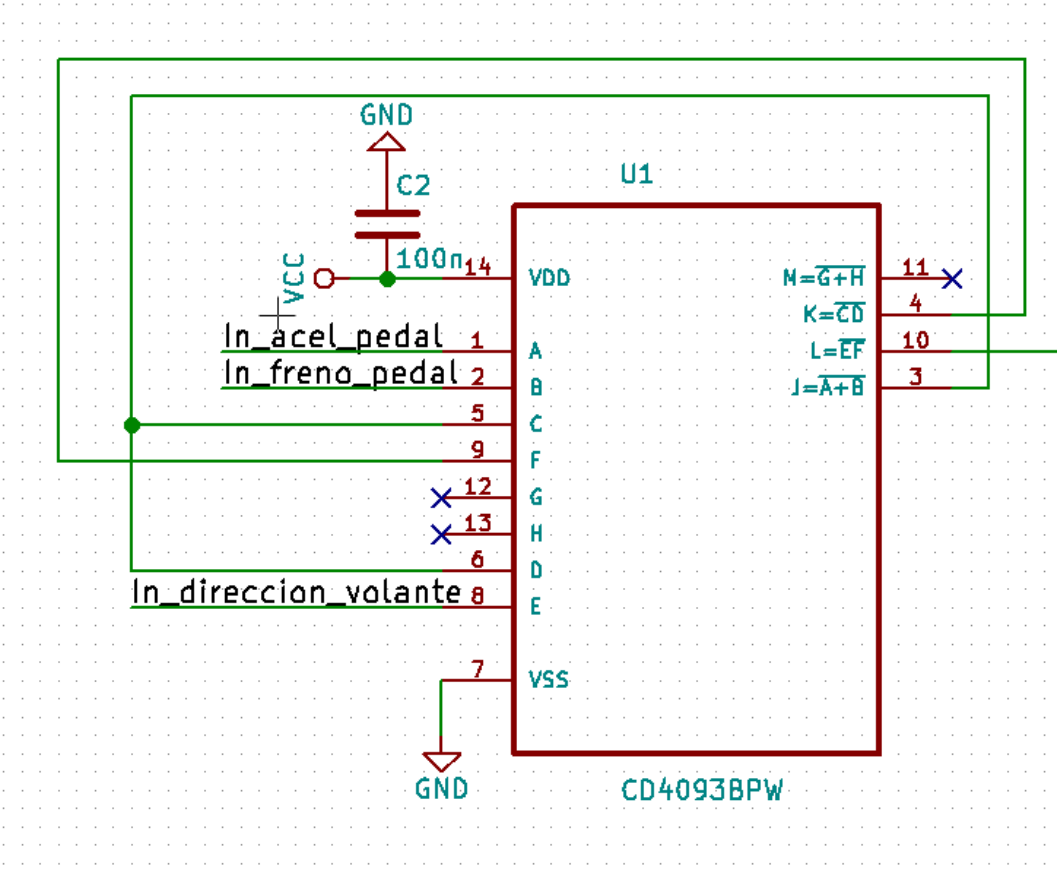


Fig. 3.8: Esquema de los Inhibits

3) Conexión del pulsador**[24]** encargado de realizar el cambio de manual a automático. Debido a que el pulsador es externo al sistema (Fig. 3.9), será necesario llevarlo al conector para poder sacar los dos cables que lo controlan. Además, se ha añadido el control anti rebotes necesario para que no se creen falsos positivos (Fig. 3.10), como era requerido por las condiciones de diseño. Dispone de un led que indicará el modo de conducción en el que nos encontramos.

Fig. 3.9: Pulsador de cambio de modo

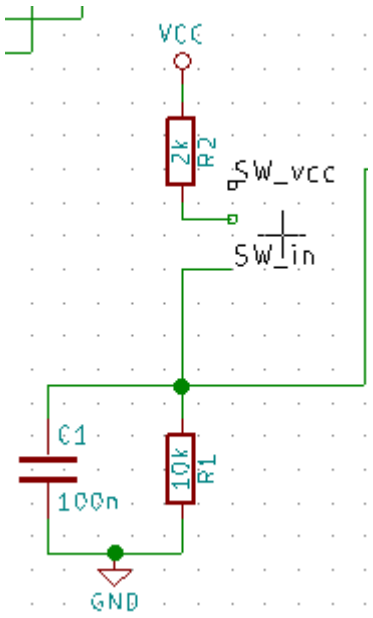


Fig. 3.10: Esquema del pulsador

4) La siguiente etapa del diseño dentro del esquemático es el biestable JK. Para ello, se ha utilizado nuevamente el dispositivo CD4093BPW, como se muestra en la Fig. 3.11.

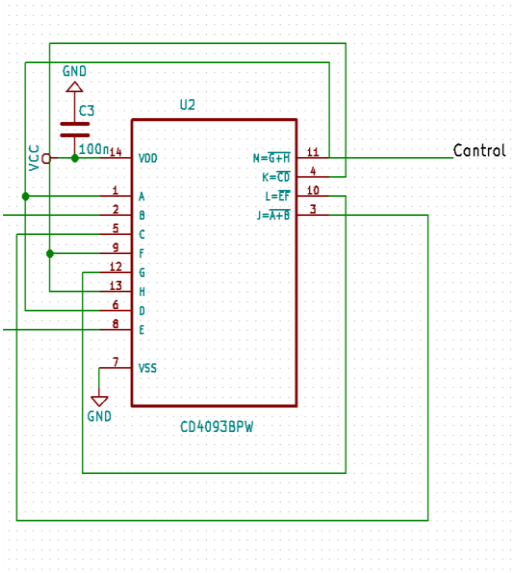


Fig. 3.11: Esquema del biestable JK

Debido a que para la realización de este componente se necesitan cuatro puertas NAND, el dispositivo utilizado es el ideal para su realización. La salida Q (Control) del biestable será la que determine si el vehículo se encuentra en modo manual (Nivel bajo, 0 voltios) o en modo automático (Nivel alto, 12 voltios). Esta salida es el pin M del dispositivo. Las entradas del dispositivo son los pines B y E. B se encuentra conectado con la salida de las puertas NAND de los Inhibit, por lo que es la entrada K del biestable, mientras que E está conectado con el pulsador, por lo que es la entrada J del biestable. Además, como en todos los dispositivos, se utilizará un condensador de desacoplo para la alimentación.

5) Una vez que ya se dispone del sistema que sea capaz de mantener la señal de automático, salvo que se realice cualquier acción por parte del conductor mediante los pedales o el volante, se debe diseñar el sistema de conmutación de la señal de salida entre la generada de forma nativa por el coche o la generada por los medios de control que han sido añadidos

al coche, dirigido por el micro controlador instalado, en función del modo de conducción en el que se encuentre el vehículo.

Las señales que deben ser multiplexadas son:

A. Acelerador (Fig. 3.12)

Mediante la señal de Control se elegirá qué señal se transmite al motor, si la generada por el pedal o la proveniente del micro controlador descrita en el capítulo anterior. Se utiliza un condensador de desacoplo para la alimentación.

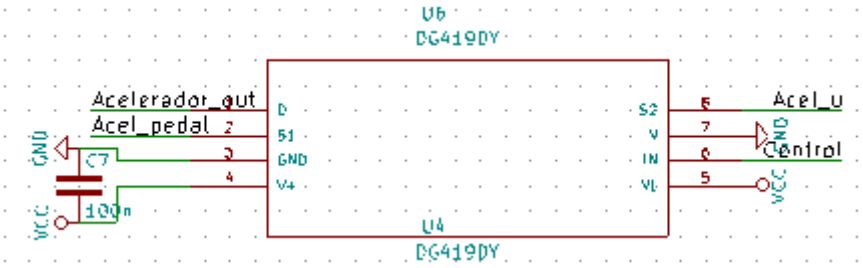


Fig. 3.12: Selector de acelerador

B. Inhibit del acelerador (Fig. 3.13)

Mediante la señal de Control se elegirá cuál se transmite al motor, si la generada por el pedal o la controlada desde el micro controlador. Para esta última, debido a la imposibilidad de generar señales mayores de 3.3 voltios, ha sido necesario hacerlo de forma externa.

Para ello se ha utilizado otro switch igual al empleado para transmitir entre las dos señales, solo que ahora debe elegir entre GND y VCC, y el pin de control será generado por el micro controlador. Por defecto, la señal del Inhibit creada para el modo automático será VCC (12 voltios), lo que significa que no está activado. Aunque se generase por error una señal en el acelerador, el Inhibit no estaría activado hasta que el micro controlador generase la señal que lo cambia. De esta manera se consigue un cierto grado de seguridad al ser necesarias dos señales para poder acelerar.

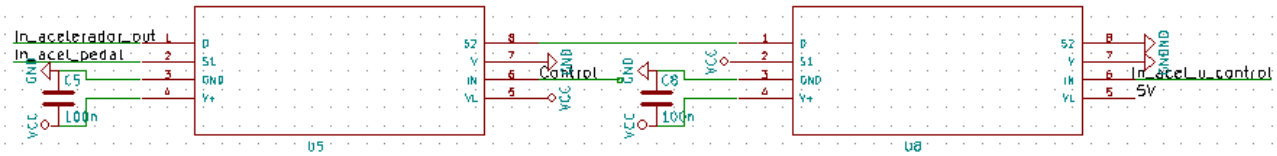


Fig. 3.13: Selector y generador del Inhibit del acelerador

C. Inhibit del freno (Fig. 3.14)

De la misma manera que con el acelerador, mediante la señal de Control se elegirá qué señal se transmite al sistema de freno, si la generada por el pedal o la proveniente del micro controlador. La realización de la señal generada por el micro controlador, debido a la imposibilidad de generar señales mayores de 3.3 voltios, ha sido necesario realizarla de forma externa.

Para ello se ha utilizado otro switch igual al empleado para transmitir entre las dos señales, solo que ahora debe elegir entre GND y VCC, y el pin de control será generado por el micro controlador. Por defecto, la señal del Inhibit creada para el modo automático será GND (0 voltios), lo que indica que está activado.

Este aspecto del diseño es un avance de cara a la futura implementación del freno automático.

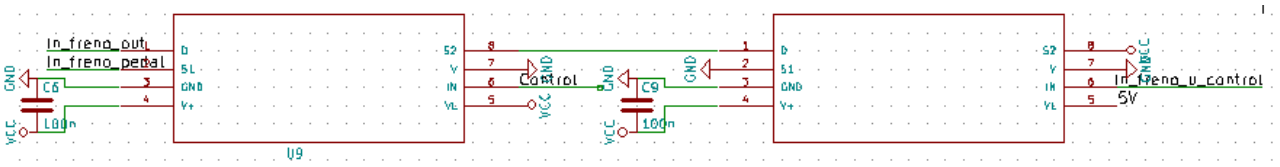


Fig. 3.14: Selector y generador del Inhibit del freno

D. Inhibit de la servodirección (Fig. 3.15)

Transmitirá la señal de activación del motor encargado de la dirección. De no encontrarse en el modo automático, la señal transmitida será circuito abierto. La señal de activación

provendrá del micro controlador. Esta señal proporciona la habilitación del motor de la servodirección en el Driver.

Fig. 3.15: Inhibit de la servodirección

E. Sentido de la dirección (Fig. 3.17)

Para su realización será necesaria la utilización de tres switches, el primero (Fig. 3.16) será el encargado de generar una señal que recree el mismo sistema implementado en el coche por medio del interruptor que se encuentra en el vehículo. La función desarrollada dependerá del sentido que envíe el micro controlador, se cortocircuitará la salida con GND, y la otra se encontrará a circuito abierto.

Fig. 3.16: Señal de sentido generada por el micro controlador

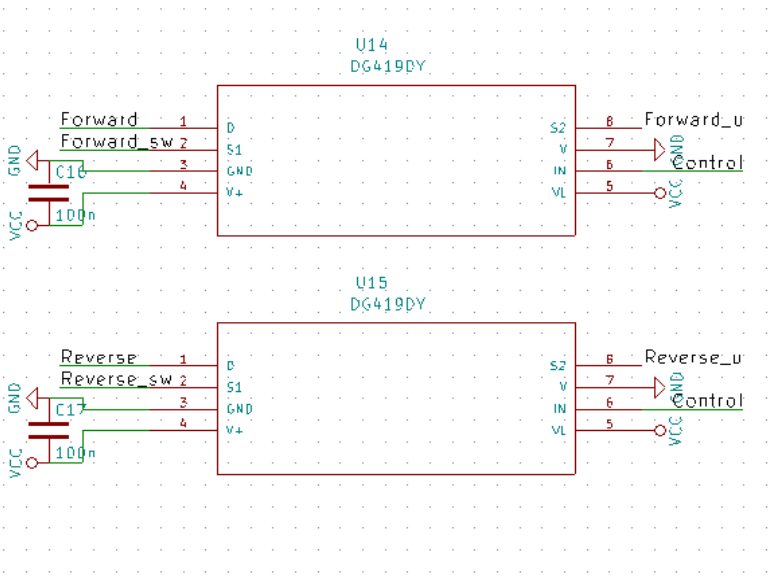


Fig. 3.17: Selector de sentido

Además, se ha implementado un sistema que permite elegir la dirección del vehículo. Este sistema debe dirigirse por la señal que controla el modo de conducción, debido a que de forma automática el sentido debe cambiarlo el micro controlador, mientras que en modo manual debe ser posible el cambio de sentido mediante el interruptor instalado en el vehículo. Este sistema, pese a estar implementado en Hardware, todavía no ha sido generado en Software.

Con esto se dará por finalizado el modo de conmutación.

6) Con el sistema de conmutación diseñado se ha aprovechado para implementar en la placa otros sistemas que estaban realizados por separado, con el objeto de reunir todas las funcionalidades de control en un mismo lugar.

   A. Para la medida suministrada al motor desde el acelerador, tanto en modo manual como automático, se ha implementado un ADC (ADS1115IDGSR) (Fig. 3.18). Es un convertidor analógico digital de 16 bits. Se comunica mediante I_2C con el micro controlador. Actualmente ya no tiene utilidad debido a que su principal función era para las pruebas iniciales. En una versión futura se usará este dispositivo para la implementación en la

servodirección de la ayuda a la conducción, permitiendo al conductor aplicar menos fuerza para realizar los giros.

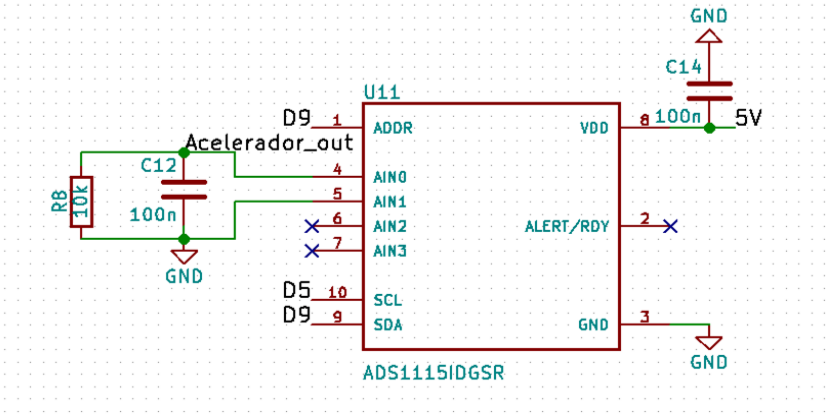


Fig. 3.18: Conexión del ADC

B. Para la generación de la señal encargada de recrear el acelerador de forma automática es necesario el uso de un DAC (MCP4725A0T) (Fig. 3.19), que es un convertidor digital analógico de 12 bits. Se utiliza el DAC debido a que el micro controlador no es capaz de suministrar señales mayores de 3.3 voltios y para conseguir una aceleración máxima en el vehículo, se ha observado experimentalmente que es necesario suministrar 4.5 voltios. El método de comunicación que empleará el micro controlador con este dispositivo es el I_2C, por lo que se han trasmitido las señales D5 y D9 para el control de éste.

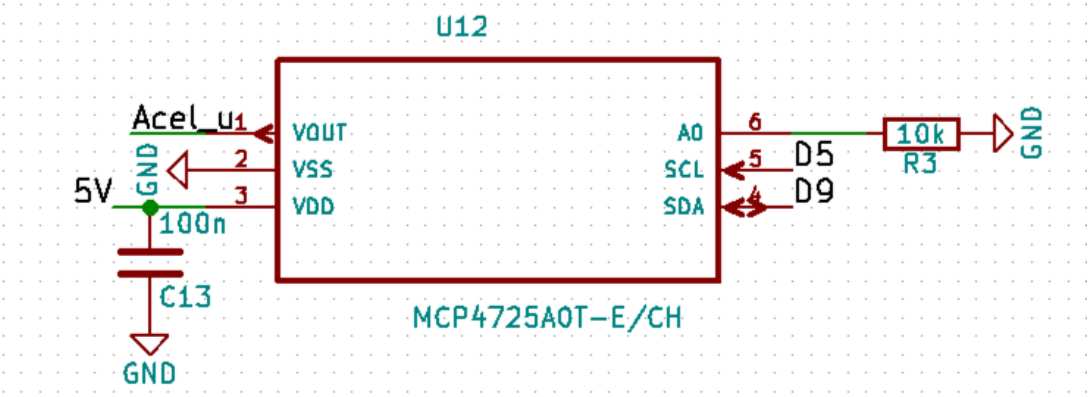


Fig. 3.19: Conexión del DAC

C. El sistema creado para la medida de la velocidad en el motor se basa en los encoders que se encuentran en éste. Para evitar posibles fallos en la medida de los encoders se ha habilitado un sistema Trigger Scmith con el fin de evitar falsos positivos (Fig. 3.20). A la salida del sistema se reduce la tensión para no ocasionar daños al micro controlador por sobretensión. Esta señal generará una interrupción en el micro controlador.

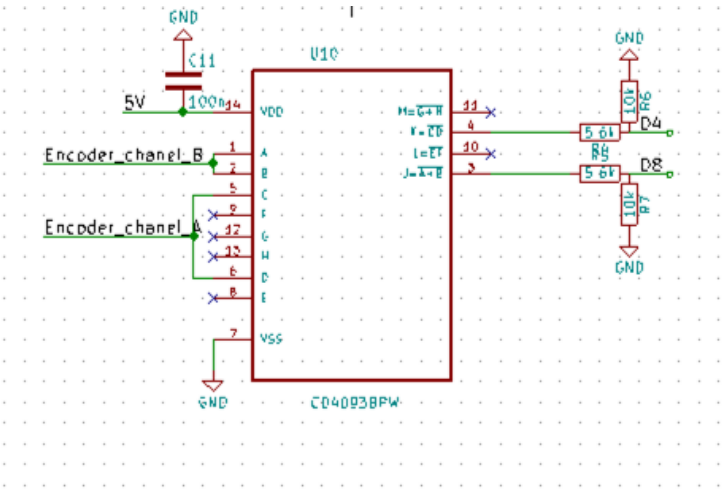


Fig. 3.20: Conexión de las puertas NAND para el Encoder del motor

7) Cuando ya disponemos de todos los elementos a controlar, debemos incluir los elementos de comunicación de la placa con el exterior. Constará de dos conectores para pines, el conector de automoción**[25]** y la posibilidad de ser conectada a una placa Arduino.

   A. Dos conectores

   El primero (Fig. 3.21) será el encargado del control de la servodirección que dirige el volante en modo automático. Éste se conectará al Driver. Las señales transmitidas serán: la alimentación de 3.3 voltios con su correspondiente masa, el Inhibit, que estará conectado a los dos enables del Driver, para habitar la PWM y, por último, las dos señales de la PWM, una realizará el giro en sentido horario y el otro en sentido anti horario.

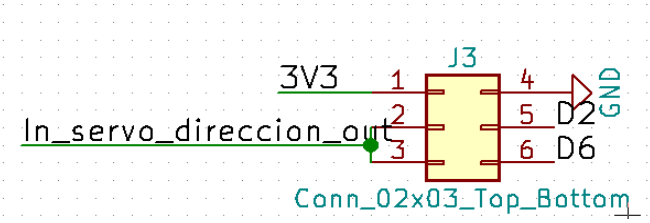


Fig. 3.21: Conector de la servodirección

El segundo (Fig. 3.22) será el encargado de conectar el sensor de ángulo con la placa. Se comunicará con el micro controlador con I_2C, por lo que deberemos vincular la señal de datos y de reloj de éste. También lo alimentaremos a 5 voltios con sus respectivas referencias a masa.

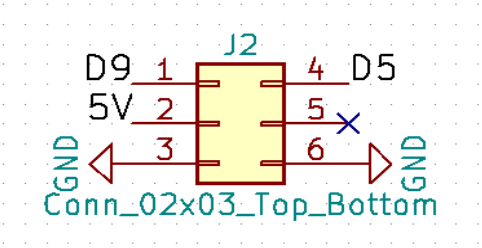


Fig. 3.22: Conector del sensor de ángulo

B. Para la conexión con la placa de Arduino se ha utilizado una huella existente en la librería Arduino-kicad-library-master**[26]**, obtenida de un repositorio online, que contiene la huella de Arduino_Uno_Shield (Fig. 3.23).

   El conexionado se ha realizado de la siguiente manera: las señales de 3.3 voltios y 5 voltios se han conectado con los elementos que necesiten esta tensión para su funcionamiento. Por otra parte, se ha procedido a la unión de las masas. Además, se ha aprovechado y se ha alimentado la placa que irá conectada con la alimentación de 12 voltios, para no depender de la alimentación actual suministrada por el USB.

También se han utilizado diferentes pines correspondientes a las GPIO de la placa para las siguientes funciones:

- **D2**
  Salida del control del servo de la dirección. Dirige un sentido. Funciona como PWM.
- **D4**
  Salida de los encoders.
- **D5**
  Datos del I_2C.
- **D6**
  Salida del control del servo de la dirección. Dirige el otro sentido. Funciona como PWM.
- **D7**
  Inhibit de la dirección del servo para el modo automático.
- **D8**
  Salida de los encoders del motor.
- **D9**
  Reloj del I_2C.
- **D10**
  Pin encargado de la generación de la señal del Inhibit del acelerador para la parte automática.
- **D11**
  Pin encargado de la generación de la señal del Inhibit del freno para la parte automática.
- **D12**
  Pin encargado de la selección del sentido en el modo automático.

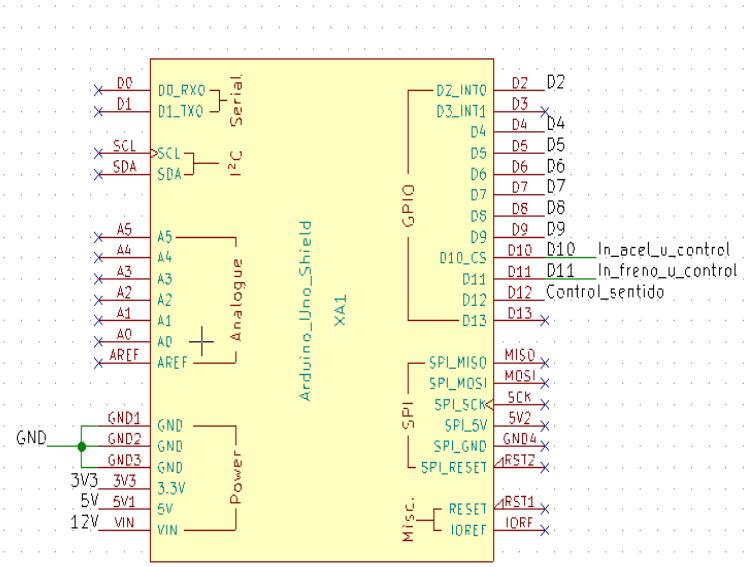


Fig. 3.23: Esquemático de la Shield de Arduino

C. Por último, se ha de utilizar un conector de 2x16 pines (Fig. 3.24) con las medidas de seguridad necesarias en automoción. La forma de crear la huella se detallará más adelante, centrando aquí la explicación en su conexionado y utilidad de cada pin. Para el esquemático se ha utilizado uno ya existente en la librería Conn de Kicad, siendo únicamente necesario vincular luego este conector con la huella que crearemos.

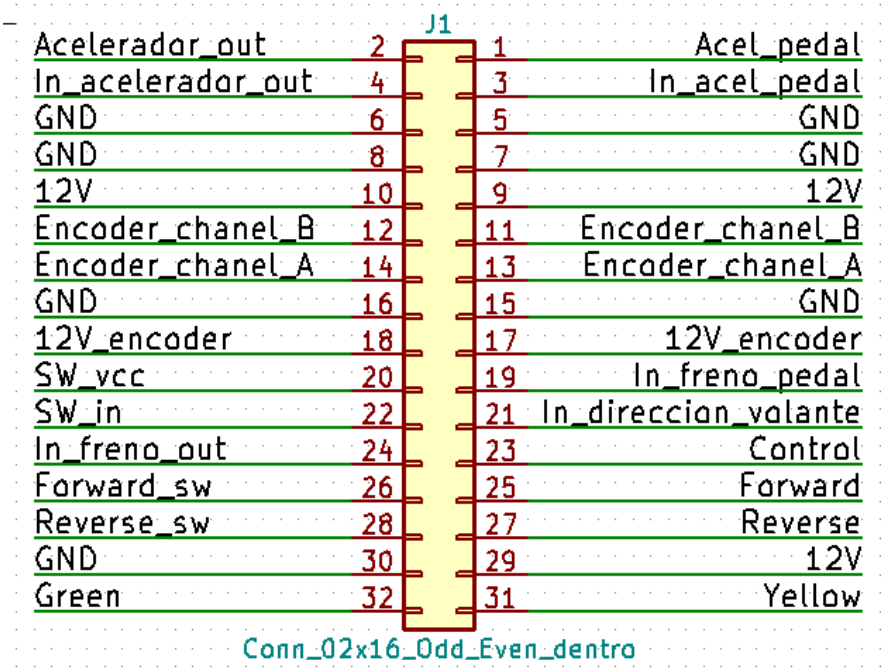


Fig. 3.24: Esquemático del conector de automoción

Salida al acelerador (Fig. 3.25)

Será la parte conectada al motor. Aquí podrán dirigirse tanto las señales generadas por el pedal de aceleración como las generadas por el micro controlador. Contiene la señal de aceleración que irá de 0.5 voltios a 4.5 voltios, así como el Inhibit del acelerador que podrá ser 12 voltios (desactivado) o 0 voltios (activado). Además, lleva dos masas y la alimentación.

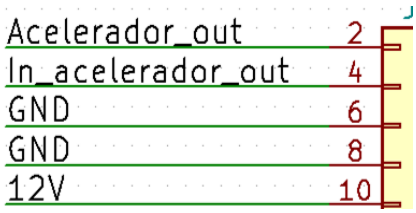


Fig. 3.25: Salidas del acelerador

Entrada del pedal del acelerador (Fig. 3.26)

Son las señales generadas por el pedal del acelerador. Lleva las mismas señales que llegan a la etapa del acelerador porque son las generadas por el pedal.

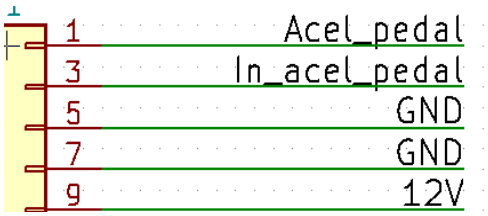


Fig. 3.26: Entradas del pedal de aceleración

Encoders del motor (Fig. 3.27)

Se encargarán de medir la velocidad del motor para conocer la velocidad del vehículo pudiendo realimentar y obtener un control más óptimo del acelerador.

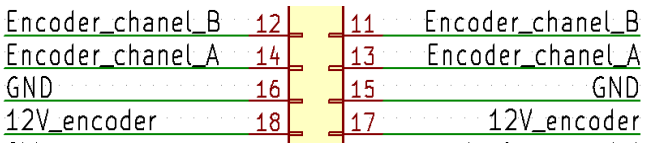


Fig. 3.27: Entradas del Encoder del motor

Pulsador (Fig. 3.28)

Señales que irán al pulsador de control de modo de conducción.

Fig. 3.28: Entradas del pulsador de cambio de modo

Inhibits (Fig. 3.29)

Los dos de la derecha son los provenientes del pedal de freno y del volante, y el de la izquierda es el que se dirige al freno, que ha sido añadido para en un futuro controlar el freno.

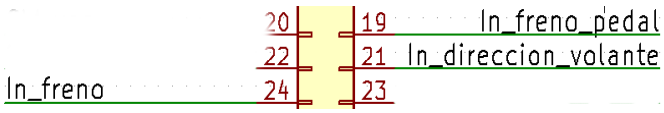


Fig. 3.29: Inhibits

Control (Fig. 3.30)

Señal que indica si estamos en modo automático (12 voltios) o manual (0 voltios). Será utilizada para los test una vez se implemente la placa. En la fase final será llevado al pulsador para encender el led instalado y así poder saber el modo de conducción en el que se encuentra el vehículo, siendo encendido modo automático y apagado en modo manual.

Fig. 3.30: Señal de control

Sentido del motor (Fig. 3.31)

Las dos de la izquierda provienen del interruptor físico que se encuentra en el vehículo, y las de la derecha serán las que se dirijan al motor una vez se haya elegido entre las generadas por el interruptor o las generadas por el sistema en modo automático.

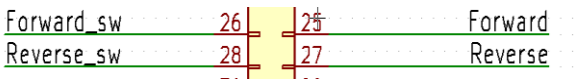


Fig. 3.31: Entradas y salidas del sentido

Alimentación (Fig. 3.32)

Para los circuitos externos, con el fin de poder realizar las pruebas necesarias para una versión 2 de la placa.

Fig. 3.32: Alimentación

Señales del sensor de torque (Fig. 3.33)

Señales del sensor de torque de la dirección para la generación del Inhibit de esta parte del sistema, y para poder diferenciar entre una fuerza generada por el motor de la servodirección y otra generada por una acción del conductor, para así poder realizar el cambio de modo cuando el conductor accione el volante.

Fig. 3.33: Señales del sensor de torque

Con el esquemático finalizado se realiza la asignación automática de componentes (Fig. 3.34). De esta manera Kicad asignará nombres a todos los elementos utilizados en el esquemático.

Fig. 3.34: Asignación de nombres al esquemático

Después, se ejecuta **CvPcb** (Fig. 3.35) para vincular cada dispositivo del esquemático con su huella, ya sea de las librerías de las que vienen pre instaladas en Kicad, las que se han añadido para este proyecto o las que se han creado debido a que no existía la huella del componente.

Para los condensadores se ha elegido el encapsulado SMD 0603, que es más pequeño de lo normal. Debido a las reducidas dimensiones de la placa es recomendable optimizar el espacio, ya que, al ser utilizados principalmente como capacidades de desacoplo, es necesario que éstas se encuentren lo más cerca posible del dispositivo.

Para los conectores se ha utilizado la distancia estándar entre pines de 2,54mm. Y para el conector de 32 pines se ha creado una huella personalizada debido a que era necesaria alguna modificación para que se adaptara al conector.

Para las resistencias se ha utilizado el encapsulado SMD 0805.

Para los siguientes componentes se han empleado las huellas obtenidas desde https://www.snapeda.com/

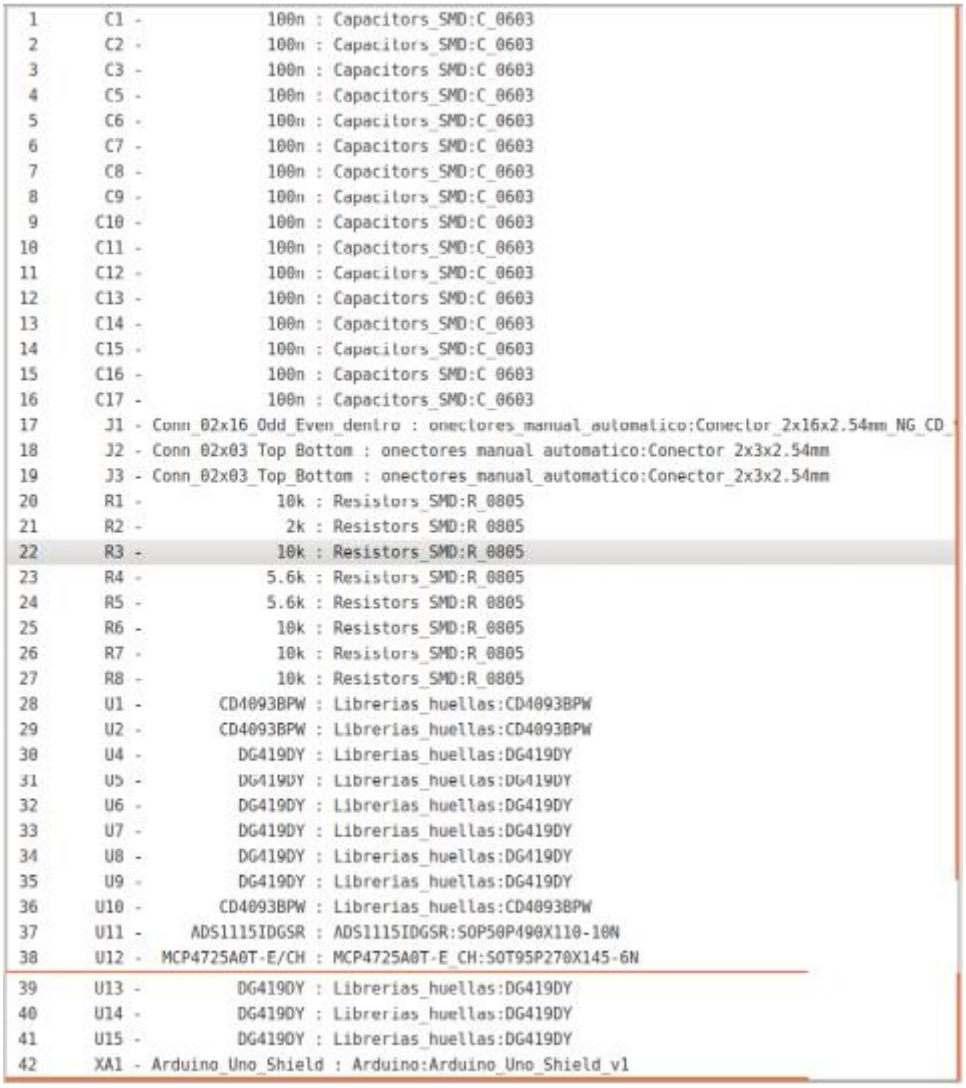
Para las puertas NAND se ha utilizado el dispositivo CD4093BPW, que es un encapsulado tipo TSSOP.

Para los switches se ha utilizado el dispositivo DG419DY, que es un encapsulado SOIC.

Para el ADC se ha utilizado el dispositivo ADS1115IDGSR.

Para el DAC se ha utilizado el dispositivo MCP4725A0T-E/CH.

Por último, para la huella de la Arduino_Shield, se ha empleado una huella modificada para la realización de esta pieza en concreto.



1	C1	100n	Capacitors_SMD:C_0603
2	C2	100n	Capacitors_SMD:C_0603
3	C3	100n	Capacitors_SMD:C_0603
4	C5	100n	Capacitors_SMD:C_0603
5	C6	100n	Capacitors_SMD:C_0603
6	C7	100n	Capacitors_SMD:C_0603
7	C8	100n	Capacitors_SMD:C_0603
8	C9	100n	Capacitors_SMD:C_0603
9	C10	100n	Capacitors_SMD:C_0603
10	C11	100n	Capacitors_SMD:C_0603
11	C12	100n	Capacitors_SMD:C_0603
12	C13	100n	Capacitors_SMD:C_0603
13	C14	100n	Capacitors_SMD:C_0603
14	C15	100n	Capacitors_SMD:C_0603
15	C16	100n	Capacitors_SMD:C_0603
16	C17	100n	Capacitors_SMD:C_0603
17	J1	Conn_02x16_Odd_Even_dentro	onectores_manual_automatico:Conector_2x16x2.54mm_NG_CD
18	J2	Conn_02x03_Top_Bottom	onectores_manual_automatico:Conector_2x3x2.54mm
19	J3	Conn_02x03_Top_Bottom	onectores_manual_automatico:Conector_2x3x2.54mm
20	R1	10k	Resistors_SMD:R_0805
21	R2	2k	Resistors_SMD:R_0805
22	R3	10k	Resistors_SMD:R_0805
23	R4	5.6k	Resistors_SMD:R_0805
24	R5	5.6k	Resistors_SMD:R_0805
25	R6	10k	Resistors_SMD:R_0805
26	R7	10k	Resistors_SMD:R_0805
27	R8	10k	Resistors_SMD:R_0805
28	U1	CD4093BPW	Librerias_huellas:CD4093BPW
29	U2	CD4093BPW	Librerias_huellas:CD4093BPW
30	U4	DG419DY	Librerias_huellas:DG419DY
31	U5	DG419DY	Librerias_huellas:DG419DY
32	U6	DG419DY	Librerias_huellas:DG419DY
33	U7	DG419DY	Librerias_huellas:DG419DY
34	U8	DG419DY	Librerias_huellas:DG419DY
35	U9	DG419DY	Librerias_huellas:DG419DY
36	U10	CD4093BPW	Librerias_huellas:CD4093BPW
37	U11	ADS1115IDGSR	ADS1115IDGSR:SOP50P490X110-10N
38	U12	MCP4725A0T-E/CH	MCP4725A0T-E_CH:SOT95P270X145-6N
39	U13	DG419DY	Librerias_huellas:DG419DY
40	U14	DG419DY	Librerias_huellas:DG419DY
41	U15	DG419DY	Librerias_huellas:DG419DY
42	XA1	Arduino_Uno_Shield	Arduino:Arduino_Uno_Shield_v1

Fig. 3.35: Asignación de huellas al esquemático

3.3.2 Diseño de huellas que no se encuentren en librerías ya hechas por terceras partes (FootPrint Editor)

Antes de finalizar el esquemático y realizar la colocación y rutado de nuestra placa de circuito impreso, es necesario añadir la huella correspondiente al conector de seguridad empleado en el diseño y la huella modificada de la Arduino Shield que va a ir conectada a la placa Olimexino-STM32. De las hojas de características del conector**[27]** se obtiene el modelo de la huella del conector (Fig. 60). Pero, previamente es necesario realizar la librería del esquemático. Para ello partimos de un conector ya incluido en las librerías básicas de Kicad y lo adaptamos a las necesidades requeridas por el diseño. Partiendo de Conn_02x16_Odd_Even [Generic connector, double row, 02x16, odd/even pin numbering scheme (row 1 odd numbers, row 2 even numbers)], debido a que es un conector de 32 pines en dos filas, se ha modificado el orden para su adaptación.

Una vez que ya se dispone de la huella del conector que se encontraba en la librería Conn, se deben realizar las modificaciones para que se adapte al conector y después vincular la huella con el esquemático.

El conector que se ha implementado en el diseño es PIN HEADER, ASSY 2X16POS, que corresponde con el de la Fig. 3.36, por lo que, atendiendo al plano del conector, se deduce que debemos seguir el modelo A. La huella deberá llevar los dos agujeros de los extremos del conector más el indicado en el modelo A.

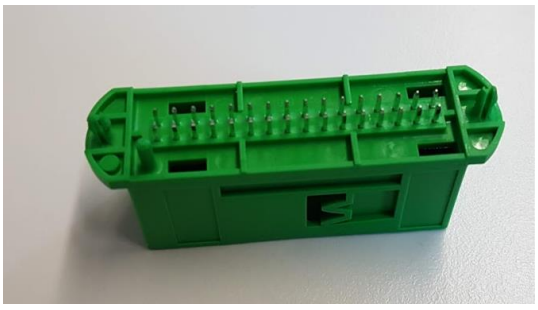


Fig. 3.36: Conector de automoción

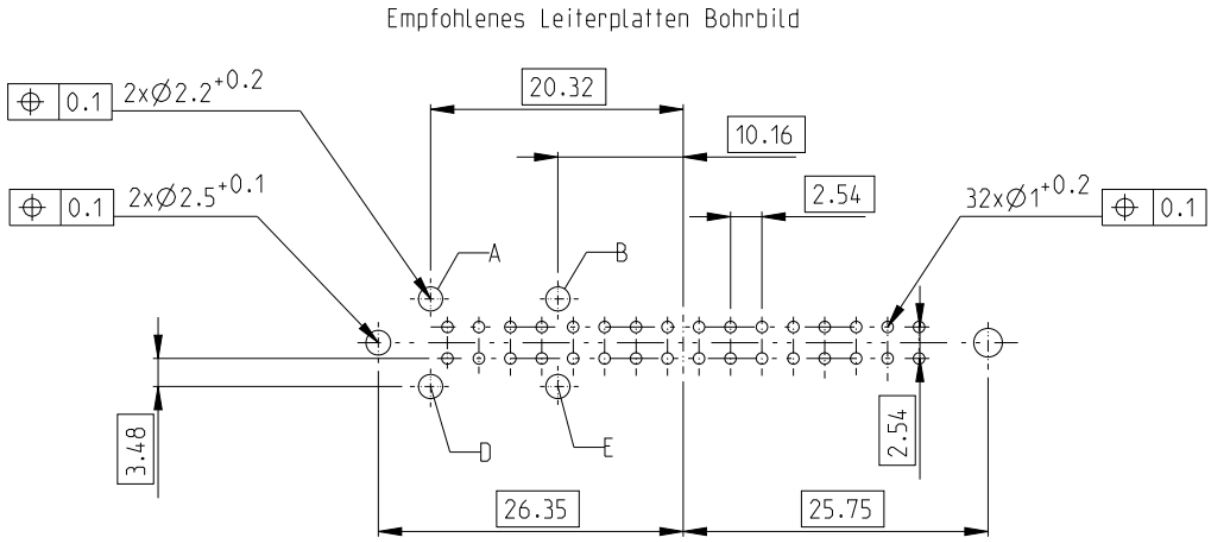


Fig. 3.37: Plano del conector de automoción

Se ha creado un Conector_2x16x2.54mm_NG_CD_966658_G1-1264110.kicad_mod que es la huella en la que se va a trabajar para conseguir que el conector adquirido con las medidas de seguridad necesarias en automoción pueda ser utilizado en el sistema.

De la hoja de características se obtienen los diámetros de los agujeros (Fig. 3.37). Los de los extremos tendrán un diámetro de 2,5 mm, mientras que el tercer agujero correspondiente al modelo A tendrá un diámetro de 2,2 mm.

Para la realización de estos agujeros es necesario crear sus pads en la huella que serán añadidos y configurados para que cuadren con el conector.

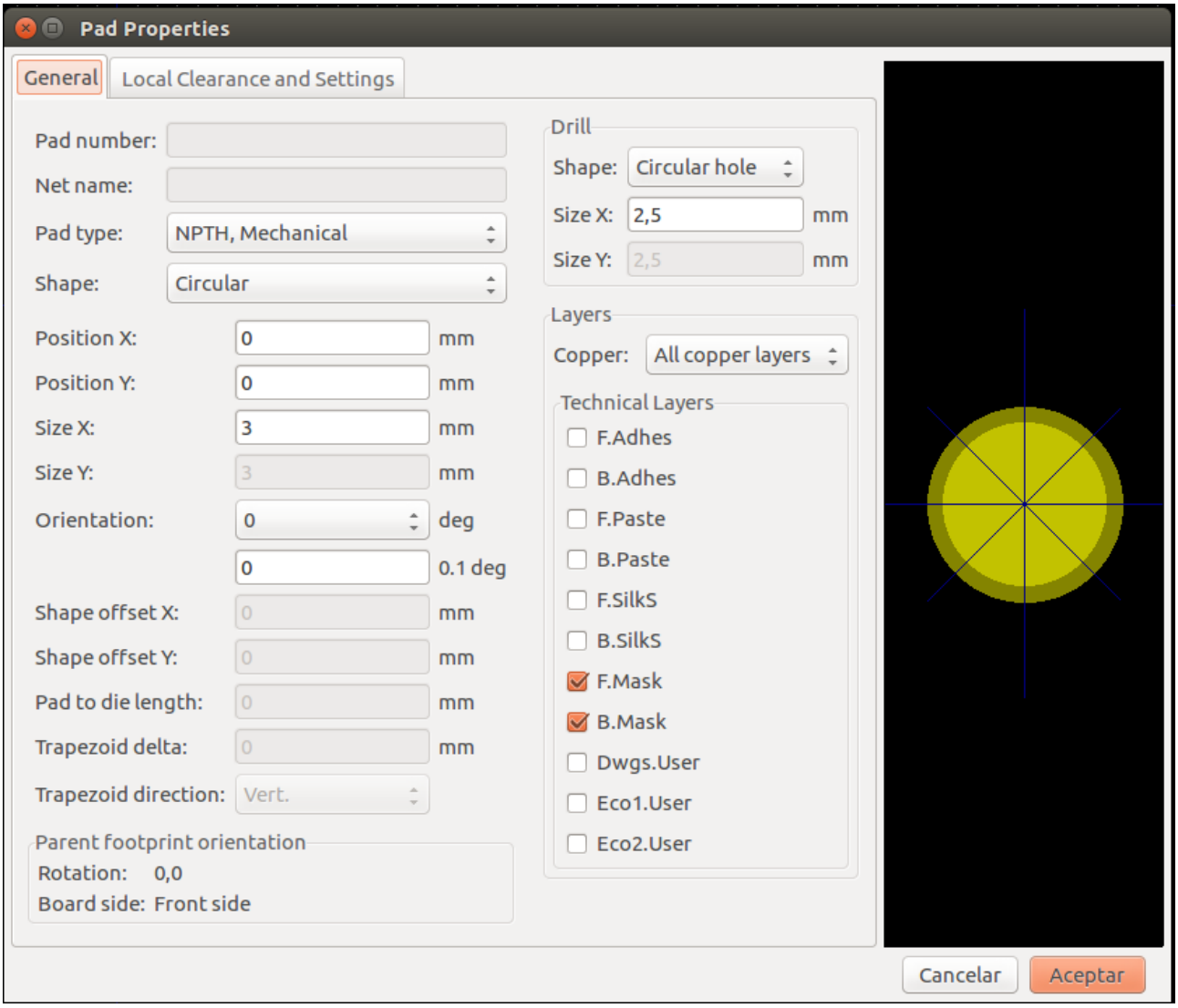


Fig. 3.38: Propiedades del agujero izquierdo

En la Fig. 3.38 se pueden apreciar los cambios necesarios para el agujero de la izquierda del conector. Inicialmente, se selecciona el tipo de pad que debe tener el agujero y su forma. El pad es NPTH, Mechanical y de forma circular. De esta manera se consigue un agujero que atraviese la placa.

Después, se debe añadir el tamaño del agujero, que es, tal como se ha especificado en el apartado de Características, de 2.5 mm de diámetro.

Continuando con el diseño, se deberá establecer la posición del agujero. Éste será colocado en el origen de coordenadas de la huella para así tener las referencias para el resto de agujeros.

Como complemento, y pensando en la implementación real del diseño, se ha añadido una capa de cobre hasta conseguir un agujero de 0,5 mm de diámetro para poder soldar el conector y que éste quede fijo a la placa de una manera más fiable.

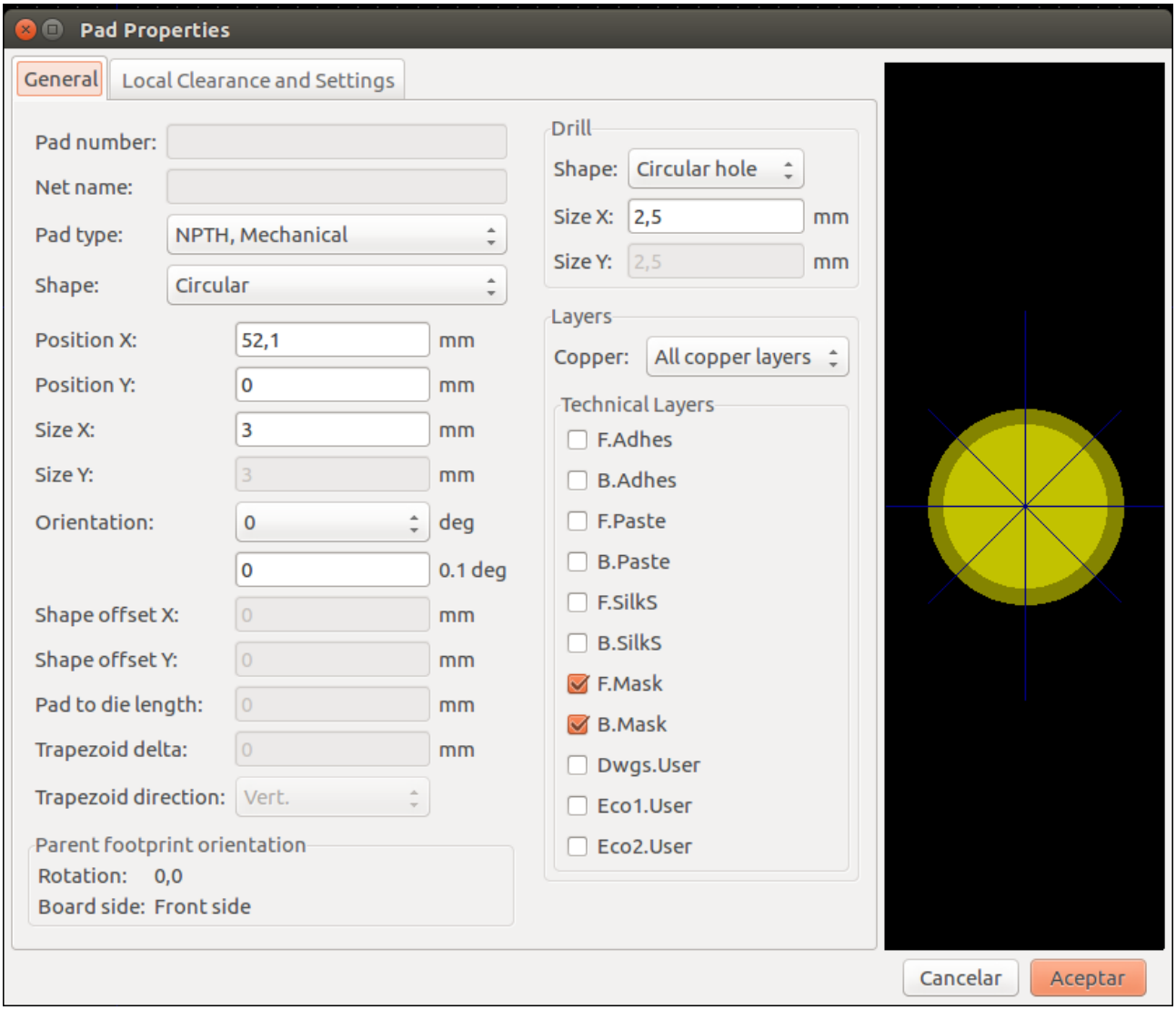


Fig. 3.39: Propiedades del agujero derecho

Para la creación de este agujero se parte del anterior (Fig. 3.38) y se cambia únicamente su posición en x para situarlo en el extremo opuesto del conector (Fig. 3.39).

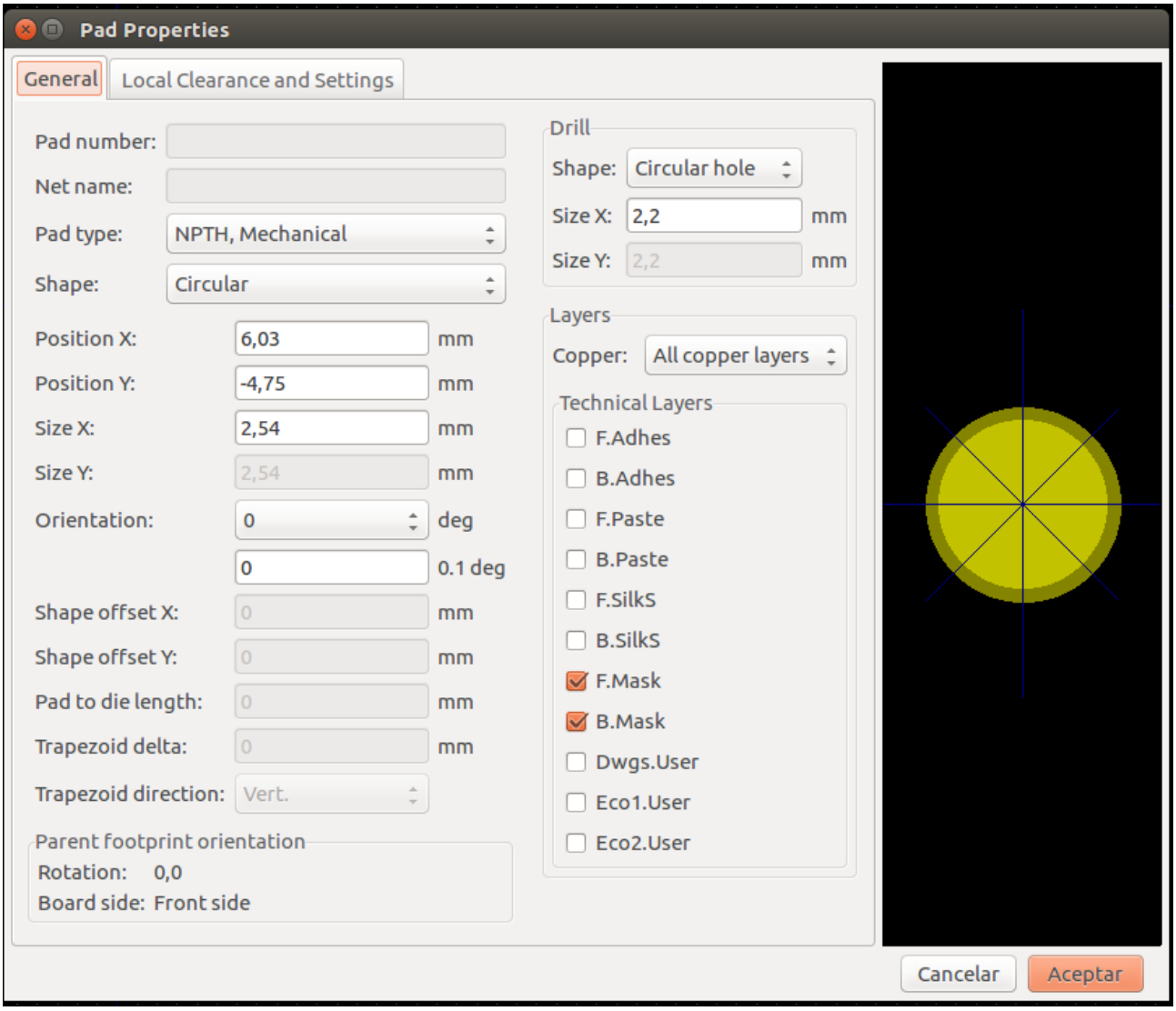


Fig. 3.40: Propiedades del agujero superior

Para el último agujero (Fig. 3.40) las diferencias con los agujeros anteriores radican en que éste tiene un diámetro de 2,2 mm, y se ha añadido cobre hasta llegar a un diámetro de 2,54 mm.

Además, se ha establecido la posición de este agujero para que cuadre con el conector.

Con esto ya se habrá terminado la huella del conector y se podrá vincular con el esquemático. El resultado final es la Fig. 3.41.

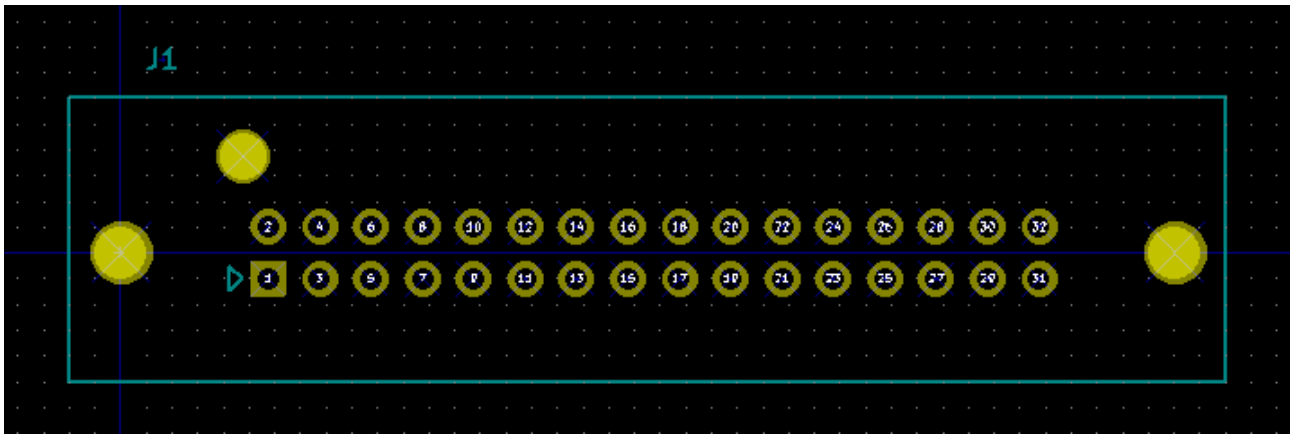


Fig. 3.41: Huella del conector de automoción

Para que la huella de la Shield de Arduino (Fig. 3.43) pueda ser conectada en la placa Olimexino-STM32 se parte de la huella descargada de la librería de Arduino para Kicad (Fig. 3.42), siendo necesario modificarla para adaptarla al diseño. Se debe suprimir el conector de 2x3 situado en la parte frontal de la huella debido a que no existe en la Olimexino-STM32. Además, se eliminarán los dos rectángulos inferiores correspondientes al USB y a la alimentación de la Arduino, que es diferente en la Olimexino-STM32.

Aunque en la Olimexino-STM32 no exista toda la fila de pines que tiene una Arduino, se mantendrán por si en un futuro decidimos cambiar a una placa Arduino.

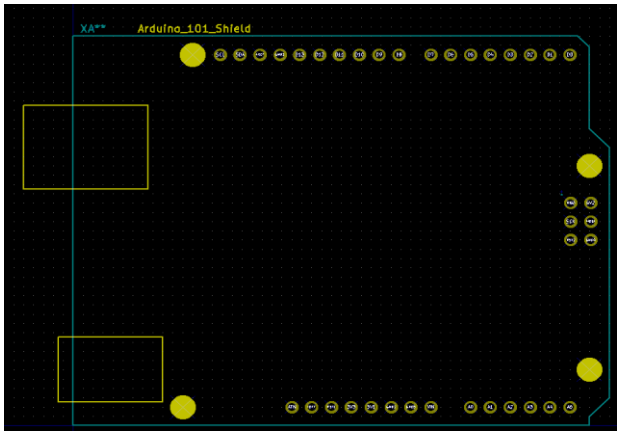


Fig. 3.42: Huella inicial de la Shield

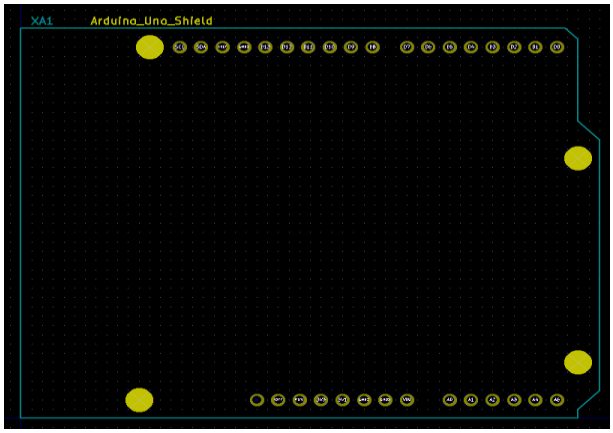


Fig. 3.43: Huella final de la Shield

Con todas las huellas asociadas con cada componente, incluidas las creadas especialmente para este diseño, es el momento de generar el archivo Netlist. Este archivo permite la transición del esquemático a la placa de circuito impreso.

Para el siguiente paso se utilizará otra aplicación de Kicad llamada Pcbnew.

3.3.3 Diseño de la placa de circuito impreso (PcbNew)

A la hora de realizar este apartado, que será el último del diseño del circuito impreso, será necesario realizar una serie de pasos para el rutado de la placa.

1. Leer la Netlist (Fig. 3.44) creada al final del apartado anterior. De esta manera todos los componentes utilizados en el esquemático se situarán en la zona de trabajo para una posterior colocación de la manera más adecuada.

Fig. 3.44: Lectura de la Netlist

2. Una vez que se dispone de todos los componentes en el área de trabajo se deben delimitar los límites de la placa. Para ello se selecciona la capa **Edge. Cuts** y, mediante la herramienta para añadir líneas o polígonos, se crear un rectángulo con las medidas requeridas.
   Para que quepa en la caja en la que irá todo el sistema de control de las ECUs, y además entren los otros sistemas que deben situarse también, se ha desarrollado con las siguientes medidas: 95,7 x 62 mm.
   Este polígono indicará al fabricante de nuestro diseño los límites de la placa, que es donde se deberá cortar la fibra de vidrio.

3. El siguiente paso consiste en colocar los componentes del diseño de una manera óptima para minimizar el uso de pistas (líneas de cobre que hacen la función de un cable) y vías (pequeños agujeros que traspasan la placa y permiten la comunicación entre las dos capas de cobre de la placa). Para ello se ha seguido un orden de colocación.
   I. Se coloca el conector de 2x16 y la huella de la Arduino_Shield modificada, debido a que deben ir en una posición concreta para el conexionado con los elementos externos.
   II. Se colocan los otros dos conectores lo más cerca posible de los pines que usen procedentes de la Arduino_Shield. Con esto se reducirán eventuales cruces de pistas, ya que al situarlos lo más cerca posible las pistas serán de menor longitud.

III. Se colocan todos los dispositivos con sus respectivas capacidades de desacoplo, lo más cerca posible de la alimentación del sistema, como se ha detallado en el apartado en el que se detalla el uso de las capacidades de desacoplo.

IV. Se ordenan los dispositivos para que todas las señales de las que dependan se encuentren lo más cerca posible.

V. Se coloca el resto de elementos necesarios y los divisores resistivos de los encoders, así como el circuito anti rebotes del pulsador.

VI. Se añade un cuadro de texto con el nombre de la placa en la capa F. Silks para que así quede impresa en la placa.

El resultado obtenido será como la Fig. 3.45.

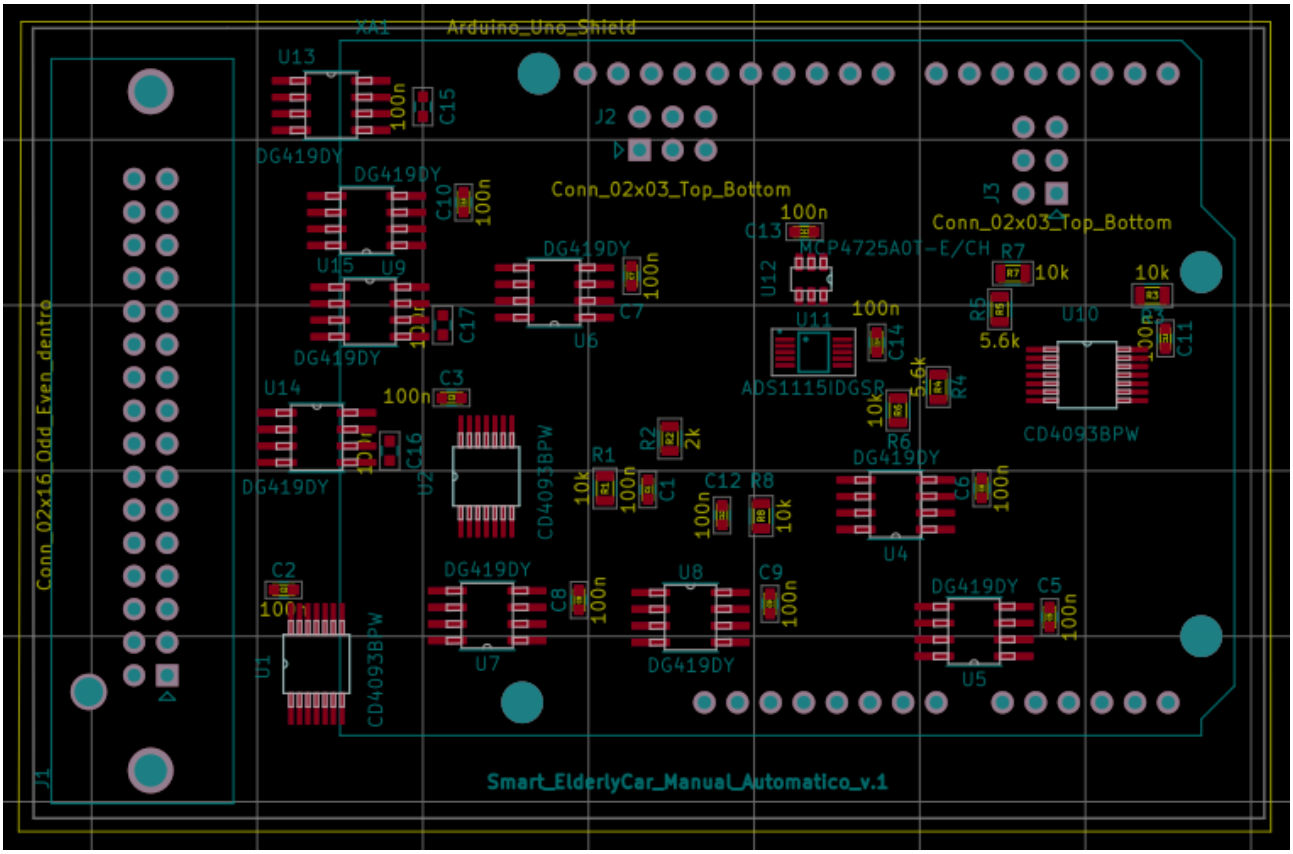


Fig. 3.45: Huella sin rutar

4. Antes de comenzar con el rutado de las pistas se deben generar las normas por las que se rigen las pistas y las vías.

   Por lo tanto, es en la pestaña de **Design Rules** donde se deben situar las normas. La manera en la que se han creado estas normas ha sido atendiendo a las capacidades de fabricación de

la empresa a la que va a ser enviada la placa. Además, se han tenido en cuenta unas limitaciones para no excederse en su precio.

Primero se han generado las normas globales del diseño, que son las mínimas que se deben cumplir (Fig. 3.46).

El primer valor indica el tamaño mínimo posible de pista. Éste se determina estableciendo el ancho de la pista: cuanto mayor sea este valor, mayor será la capacidad resistiva de la pista y menor su capacidad inductiva, pudiendo crear posibles errores en nuestras señales.

El siguiente valor es el diámetro mínimo de las vías, que, como se ha explicado anteriormente, permiten la comunicación entre pistas de diferentes capas. La mayoría de placas de circuito impreso disponen de dos capas de cobre. Éstas son pistas que no se cortocircuitarán aunque se crucen, debido a que se encuentran en niveles diferentes. Para poder cortocircuitarlas se usan las vías -un agujero que perfora las dos capas y lleva añadido cobre para poder hacer la unión-. De esta manera se pueden realizar más conexiones, aunque es recomendable utilizar las menos posibles.

Por último, se tiene en cuenta el diámetro del agujero que se creará para la vía, siendo la diferencia entre éste y el valor anterior la cantidad de cobre que se va utilizar en la vía.

Los valores correspondientes a las micro vías se dejan a 0, ya que no se van a emplear en este diseño.

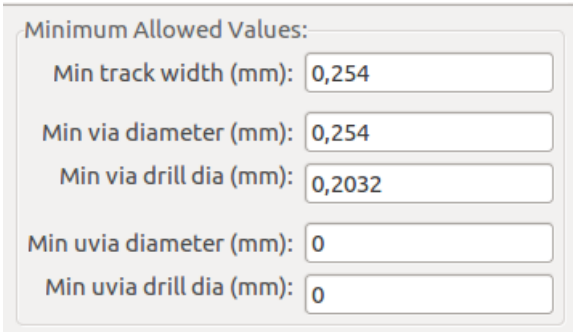


Fig. 3.46: Normas mínimas de rutado

Se define el tamaño por defecto de las pistas para la gran mayoría de vías (Fig. 3.47). Además de los valores explicados anteriormente, se debe añadir la distancia mínima entre pistas. El resto de valores se han ampliado hasta llegar al doble de los establecidos en los valores mínimos.

	Clearance	Track Width	Via Dia	Via Drill	uVia Dia	uVia Drill
Default	0,2032	0,508	0,635	0,4	0,508	0,254

Fig. 3.47: Dimensiones de las pistas por defecto

Ahora se deben limitar algunas señales, ya que parte de los encapsulados utilizados son demasiado pequeños para las dimensiones establecidas por defecto. Para ello se ha creado la clase ADC (Fig. 3.48) que lleva unos tamaños más pequeños de pistas, lo que permite que pueda llegar a un pad del encapsulado sin tener conflicto con los pads situados a los lados, ni con las pistas que llegan a éstos.

Las pistas que ha sido necesario reducir son las siguientes:

- Todas las correspondientes al dispositivo ADC (ADS1115IDGSR).
- Algunas de las que llegan a los pads de las puertas NAND (CD4093BPW), ya que a la hora de realizar el rutado de la placa no se podía ejecutar el rutado de estas pistas porque tocaban las de su alrededor.

ADC	0,2032	0,254	0,635	0,4	0,508	0,254

Fig. 3.48*:* Dimensiones de las pistas de tamaño reducido

5. Con los componentes situados en su posición más óptima y las normas de diseño ya generadas se procederá al rutado de la placa.

   Debido a la cantidad de pads y pistas necesarias para la creación de todo el circuito, se ha utilizado una herramienta externa para el rutado de la placa, de forma que se realice de manera automática. Se ha logrado un rutado óptimo minimizando el uso de vías y el tamaño de las pistas.

   Para la instalación de esta herramienta para el sistema operativo Ubuntu es necesario instalar una serie de programas externos a Kicad:

   1. En primer lugar, debido a que es una herramienta dependiente de Java, se utilizará el NetBeans. Para su instalación debemos seguir una serie de pasos.

A. En un terminal se ejecutan los siguientes comandos para primero instalar la versión 8 del JDK de Oracle
   A. **sudo add-apt-repository ppa:webupd8team/java**
   B. **sudo apt-get update**
   C. **sudo apt-get install oracle-java8-installer**

B. Se instala Netbeans
   A. Se descarga de la web oficial. https://netbeans.org/downloads/
   B. En un terminal se ejecutan los siguientes comandos:
      - chmod +x <nombre del archivo>. En este caso será **chmod +x netbeans-8.2-javase-linux.sh**
      - ./ <nombre del archivo>. En este caso será **./ netbeans-8.2-javase-linux.sh**
   C. Se abre una pestaña nueva en la que se debe indicar dónde se encuentra el archivo del JDK 8: en la carpeta usr/java.

2. Con el Netbeans instalado, se debe instalar el git. Para ello se ejecuta:
   sudo apt-get install git
3. Después hay que dirigirse a una carpeta en la que se desee guardar los archivos necesarios para el Freerouting desde el terminal con el comando cd, y, una vez situados en ella, se realiza la siguiente instrucción:
   git clone https://github.com/nikropht/FreeRouting.git
4. Por último, antes de crear el proyecto, se debe obtener el archivo .jar, que se obtiene de: https://github.com/kellertuer/Gravel/blob/master/javahelp/jh2.0/javahelp/lib/jh.jar
   Y se copia en la carpeta anterior.
5. Ahora hay que dirigirse al Netbeans y crear un nuevo proyecto.
   A. Se crea un proyecto nuevo.
   B. Se selecciona que sea un proyecto Java con fuentes ya existentes.
   C. Se añade la carpeta creada en los pasos anteriores como el paquete de fuentes.
   D. Se compila el proyecto.
   E. Con el botón derecho del ratón en el nombre del proyecto hay que dirigirse a las propiedades.

F. En la pestaña de Librerías se añade como JAR el archivo descargado anteriormente.
G. En la pestaña Web Start se marca la casilla para habilitarlo.
H. Se compila el proyecto otra vez. Con ello ya se dispondrá del Autorouting.

6. Otra vez de nuevo en Kicad, se selecciona la Fig. 3.49 y se exporta el diseño como un .dsn

Fig. 3.49: Exportar diseño al Autorouting

7. Se ejecuta el proyecto creado desde Netbeans y se abre esta pestaña Fig. 3.50.

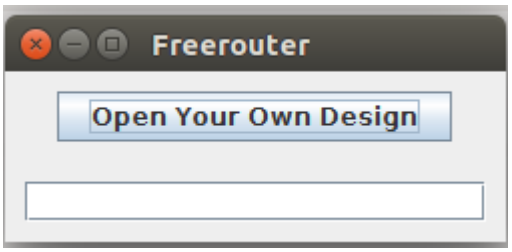


Fig. 3.50: Autorouting

8. Se abre el archivo con extensión *.dsn* creado desde Kicad y aparecerá el PCB con la posición y las pistas que se hayan rutado manualmente. Éstas han sido las que generaban conflictos debido al tamaño de la pista.
9. Se ejecuta el Autorouter (Fig. 3.51).

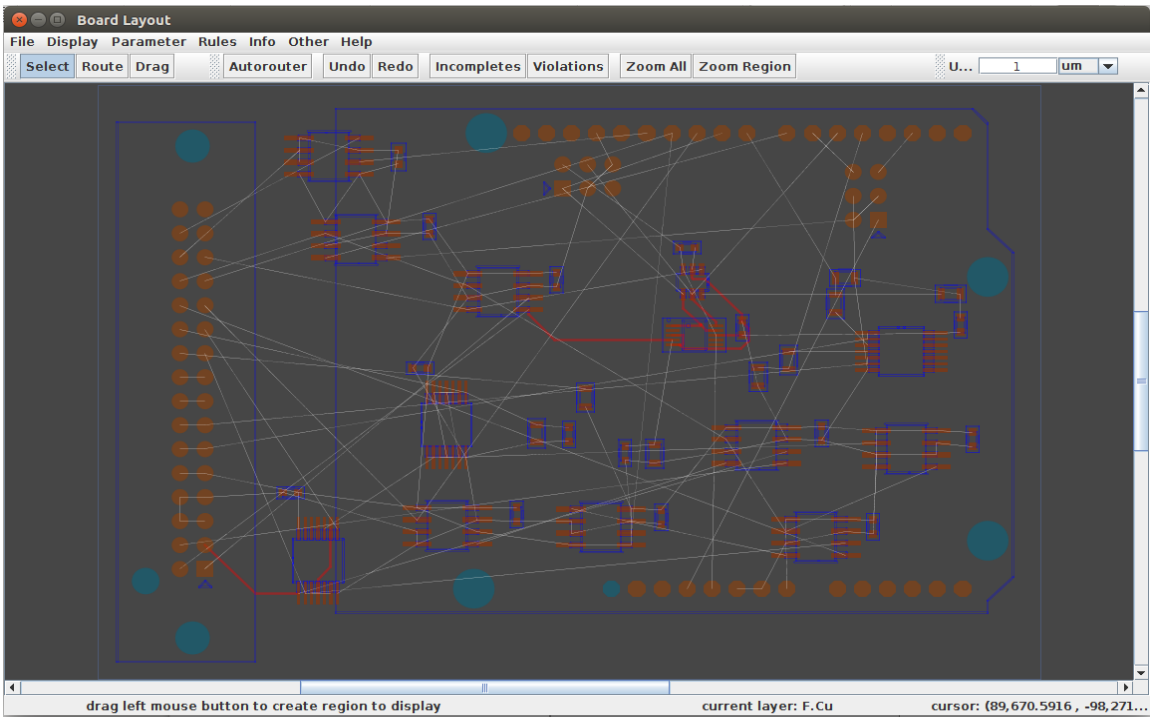


Fig. 3.51: Diseño en el Autorouting sin rutar

10. Una vez haya acabado, se obtendrá la Fig. 3.52. y se podrá exportar el archivo Specctra de vuelta a Kicad, donde se consigue la Fig. 3.53.

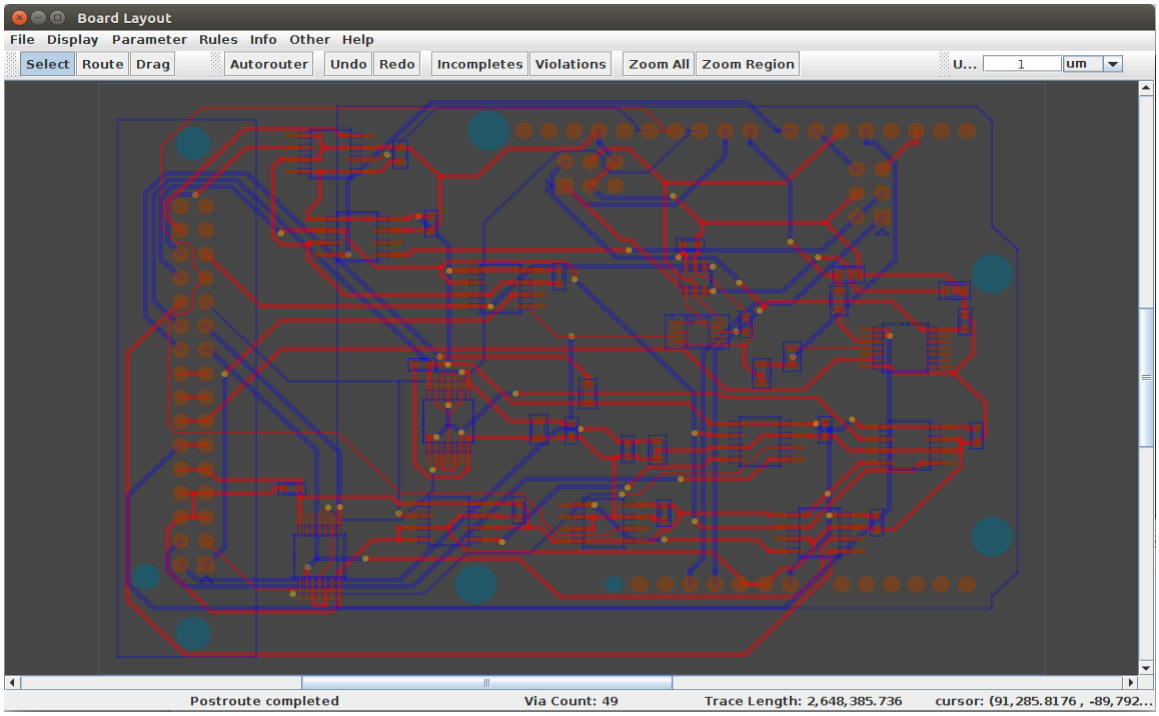


Fig. 3.52 Diseño en el Autorouting rutado

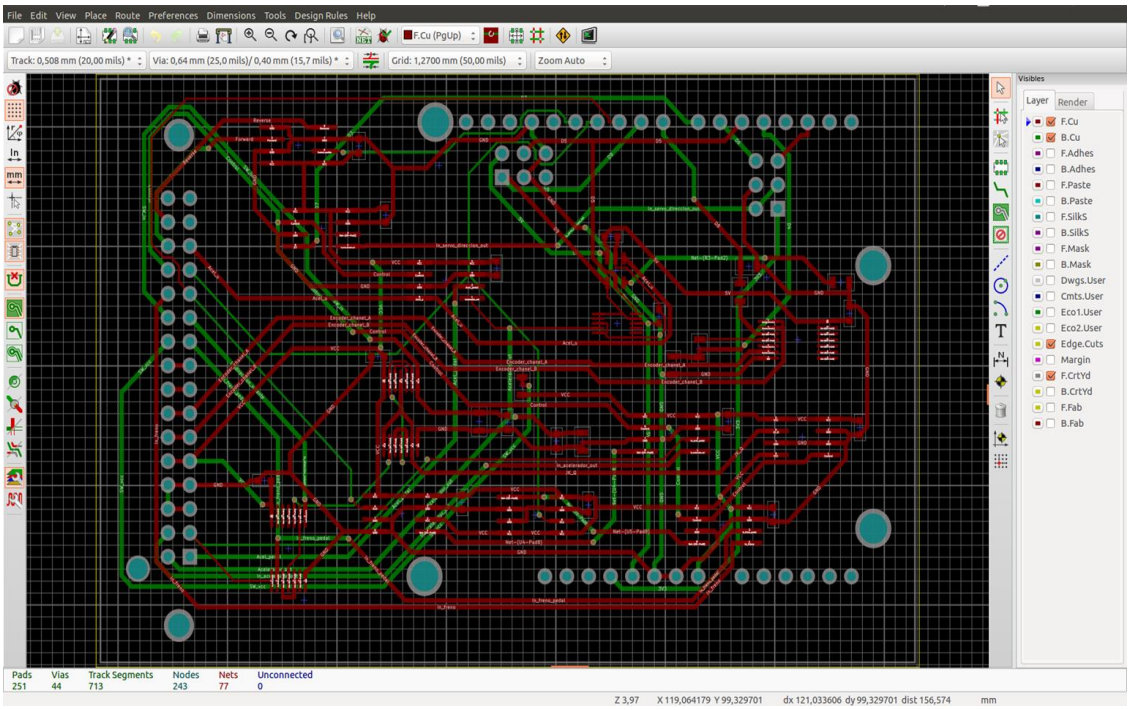


Fig. 3.53: Diseño en Kicad rutado

11. Se revisa que la placa se encuentre acorde a las características requeridas. Ya es momento de pasar a la última fase: la fabricación.

3.4 Fabricación y montaje de la placa

Con el diseño ya terminado, es momento de enviar el sistema a fabricar. Debido a que son necesarias unas técnicas especiales para realizarlo, se enviará a una empresa externa para que lo fabrique.

En este caso, se ha enviado a Eurociruits**[28]**, una empresa europea que se encarga de la fabricación de circuitos impresos.

Para una aproximación del presupuesto y que no haya problemas de diseño, Eurocircuits tiene una herramienta que permite analizar el diseño y genera un presupuesto aproximado, comprobando que se cumplan todas las limitaciones de diseño que previamente se han configurado para que se pueda realizar la fabricación de la placa sin problemas.

Una vez que la ha analizado y comprobado que todos los valores son correctos, debería aparecer la información de la Fig. 3.54 y se podrá realizar el pedido de la placa. De no ser así, se debe entrar en el esquema detallado y ajustar el diseño, debido a que algunas de las condiciones de diseño impuestas no pueden ser realizadas por imposibilidades técnicas de la fábrica.

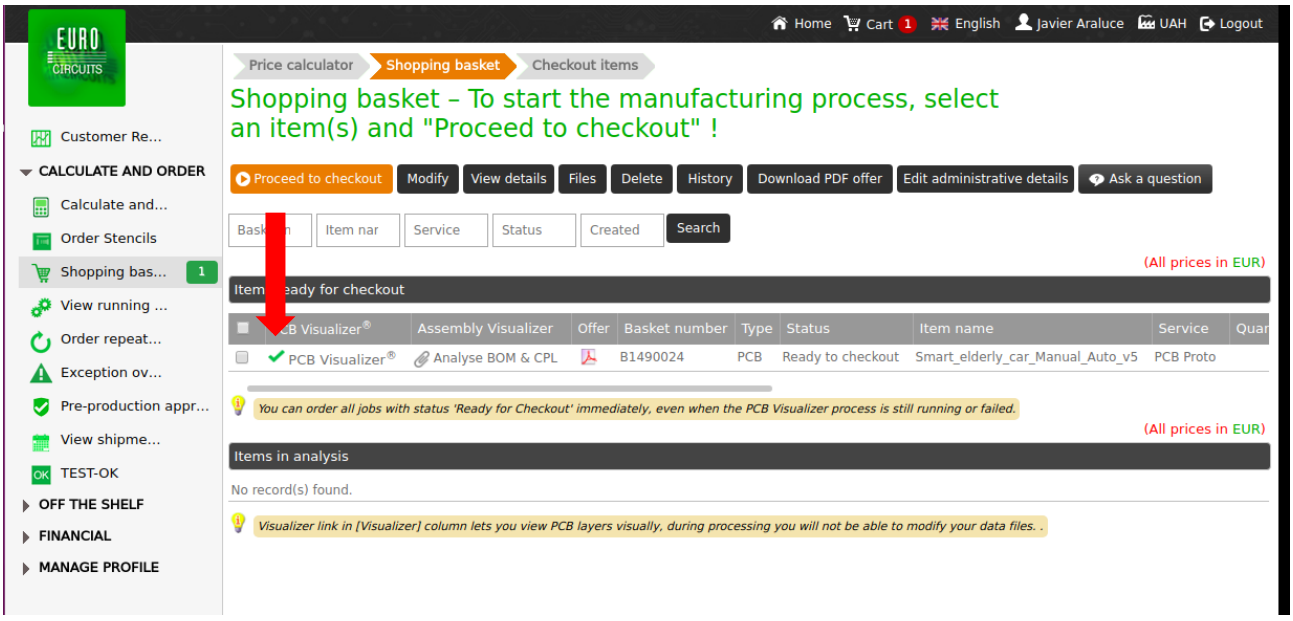


Fig. 3.54: Pedido de la placa

Con la placa ya fabricada, es momento de soldar todos los componentes en ella. Para ello generamos la lista de materiales (Fig. 3.55) desde Kicad con el fin de pedirlos en cualquier tienda de electrónica.

Manual_auto.xml

```
Reference, Value, Footprint, Datasheet, Field4, Field5, Field6, Field7, Field8

R1,10k,Resistors_SMD:R_0805,
R2,2k,Resistors_SMD:R_0805,
R3,10k,Resistors_SMD:R_0805,
R4,5.6k,Resistors_SMD:R_0805,
R5,5.6k,Resistors_SMD:R_0805,
R6,10k,Resistors_SMD:R_0805,
R7,10k,Resistors_SMD:R_0805,
R8,10k,Resistors_SMD:R_0805,


C1,100n,Capacitors_SMD:C_0603,
C2,100n,Capacitors_SMD:C_0603,
C3,100n,Capacitors_SMD:C_0603,
C5,100n,Capacitors_SMD:C_0603,
C6,100n,Capacitors_SMD:C_0603,
C7,100n,Capacitors_SMD:C_0603,
C8,100n,Capacitors_SMD:C_0603,
C9,100n,Capacitors_SMD:C_0603,
C10,100n,Capacitors_SMD:C_0603,
C11,100n,Capacitors_SMD:C_0603,
C13,100n,Capacitors_SMD:C_0603,
C12,100n,Capacitors_SMD:C_0603,
C14,100n,Capacitors_SMD:C_0603,
C15,100n,Capacitors_SMD:C_0603,
C16,100n,Capacitors_SMD:C_0603,
C17,100n,Capacitors_SMD:C_0603,

U1,CD4093BPW,Librerias_huellas:CD4093BPW,CMOS Quad 2-Input NAND Schmitt Triggers 14-TSSOP -55 to 125,TSSOP-14 Texas Instruments,Texas Instruments,0.29 USD,CD4093BPW,Good
U2,CD4093BPW,Librerias_huellas:CD4093BPW,CMOS Quad 2-Input NAND Schmitt Triggers 14-TSSOP -55 to 125,TSSOP-14 Texas Instruments,Texas Instruments,0.29 USD,CD4093BPW,Good
U10,CD4093BPW,Librerias_huellas:CD4093BPW,CMOS Quad 2-Input NAND Schmitt Triggers 14-TSSOP -55 to 125,TSSOP-14 Texas Instruments,Texas Instruments,0.29 USD,CD4093BPW,Good

U6,DG419DY,Librerias_huellas:DG419DY,DG419DY,None,Vishay Siliconix,SO-8 Vishay,Unavailable,DG419 Series SPDT 15 V 20 Ohm Precision Monolithic CMOS Analog Switch - SOIC-8N
U7,DG419DY,Librerias_huellas:DG419DY,DG419DY,None,Vishay Siliconix,SO-8 Vishay,Unavailable,DG419 Series SPDT 15 V 20 Ohm Precision Monolithic CMOS Analog Switch - SOIC-8N
U4,DG419DY,Librerias_huellas:DG419DY,DG419DY,None,Vishay Siliconix,SO-8 Vishay,Unavailable,DG419 Series SPDT 15 V 20 Ohm Precision Monolithic CMOS Analog Switch - SOIC-8N
U8,DG419DY,Librerias_huellas:DG419DY,DG419DY,None,Vishay Siliconix,SO-8 Vishay,Unavailable,DG419 Series SPDT 15 V 20 Ohm Precision Monolithic CMOS Analog Switch - SOIC-8N
U5,DG419DY,Librerias_huellas:DG419DY,DG419DY,None,Vishay Siliconix,SO-8 Vishay,Unavailable,DG419 Series SPDT 15 V 20 Ohm Precision Monolithic CMOS Analog Switch - SOIC-8N
U9,DG419DY,Librerias_huellas:DG419DY,DG419DY,None,Vishay Siliconix,SO-8 Vishay,Unavailable,DG419 Series SPDT 15 V 20 Ohm Precision Monolithic CMOS Analog Switch - SOIC-8N
U13,DG419DY,Librerias_huellas:DG419DY,DG419DY,None,Vishay Siliconix,SO-8 Vishay,Unavailable,DG419 Series SPDT 15 V 20 Ohm Precision Monolithic CMOS Analog Switch - SOIC-8N
U14,DG419DY,Librerias_huellas:DG419DY,DG419DY,None,Vishay Siliconix,SO-8 Vishay,Unavailable,DG419 Series SPDT 15 V 20 Ohm Precision Monolithic CMOS Analog Switch - SOIC-8N
U15,DG419DY,Librerias_huellas:DG419DY,DG419DY,None,Vishay Siliconix,SO-8 Vishay,Unavailable,DG419 Series SPDT 15 V 20 Ohm Precision Monolithic CMOS Analog Switch - SOIC-8N

U11,ADS1115IDGSR,ADS1115IDGSR:SOP50P490X110-10N,Texas Instruments,296-38849-1-ND,ADS1115IDGSR,MSOP-10 Texas Instruments,16-Bit ADC with Integrated MUX, PGA, Comparator, Oscillato

U12,MCP4725A0T-E/CH,MCP4725A0T-E_CH:SOT95P270X145-6N,MCP4725 Series 1 Ch I2C EEPROM Memory 5.5 V 12-Bit D/A Converter SMT - SOT-23-6,Microchip,Good,0.92 USD,MCP4725A0T-E/CH,SOT-2

//J1,Conn_02x16_Odd_Even_dentro,onectores_manual_automatico:Conector_2x16x2.54mm_NG_CD_966658_G1-1264110,
//J2,Conn_02x03_Top_Bottom,onectores_manual_automatico:Conector_2x3x2.54mm,
//J3,Conn_02x03_Top_Bottom,onectores_manual_automatico:Conector_2x3x2.54mm,

//XA1,Arduino_Uno_Shield,Arduino:Arduino_Uno_Shield_v1,
```

Fig. 3.55: Lista de materiales

3.4.1 Montaje de la placa

Una vez se dispone de todos los componentes y de la placa de circuito impreso (Fig. 3.56), es momento de soldar todos los componentes en ella.

Para que sea más cómodo realizar las soldaduras se debe llevar un orden de soldadura, empezando por los componentes más complicados, que en este caso son el ADS1115IDGSR y CD4093BPW, debido a que la distancia entre patillas es la más reducida de todos los dispositivos. Después se continua con el MCP4725A0T, para seguir con los DG419DY, con esto ya se tendrían todas las puertas conectadas. Ahora es momento de soldar las resistencias y condensadores, y, por último, los conectores, ya que, si se sueldan éstos en primer lugar, no habrá una superficie estable en la que poder trabajar y será más tedioso soldar los otros componentes. Para las primeras soldaduras es recomendable hacer uso del microscopio que permita una mejor visión.

Con la placa montada (Fig. 3.57), hay que realizar los cables necesarios para su conexionado con el coche. Para ello se han utilizado dos mangueras de 14 hilos cada una. Aquí estarán la mayoría de señales del conector de automoción, exceptuando las que van al pulsador que, al estar más cerca de la placa, no necesita un cable tan largo. Además, se utilizarán dos cables de 6 hilos cada uno de una longitud determinada para la servodirección y el sensor de ángulo de ésta.

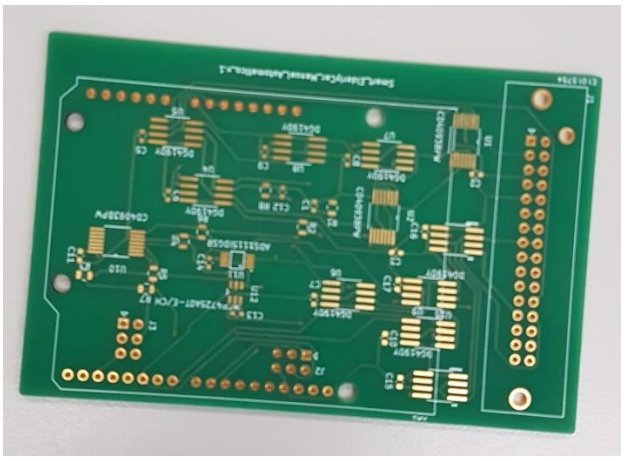


Fig. 3.56: Placa vacía

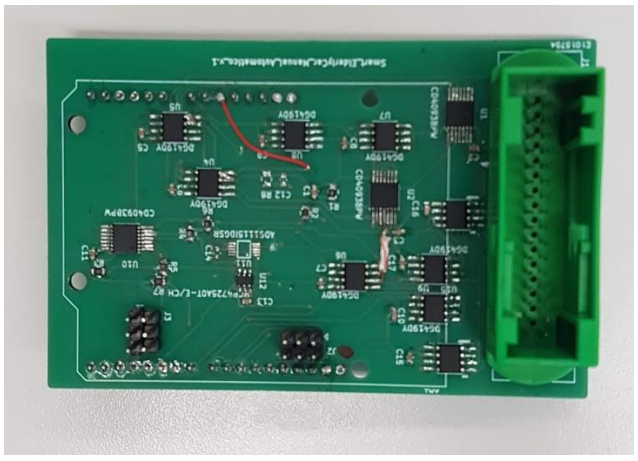


Fig. 3.57: Placa montada

3.5 Modificaciones

Con la placa ya montada es momento de probar todas sus funciones y, de ser necesario, realizar alguna modificación del diseño original para que funcione como es debido.

El primer problema encontrado es que, por defecto, se inicia en modo automático debido a gliches de las puertas, por lo que se ha realizado un circuito secundario capaz de ejecutar un reset asíncrono. Para ello se ha montado el circuito (Fig. 3.58) que recreará el accionamiento de uno de los pedales durante el inicio del sistema, quedándose así en modo manual hasta que active el pulsador encargado de pasar a modo automático. La manera de recrearlo es mediante una red RC. El condensador necesitará cargarse, y mientras éste se carga el Inhibit estará recibiendo 0 voltios, por lo que es como si estuviera pisado. El tiempo de carga con estos valores es de un segundo, siendo más que suficiente para realizar su función y que no interfiera con el resto de tareas.

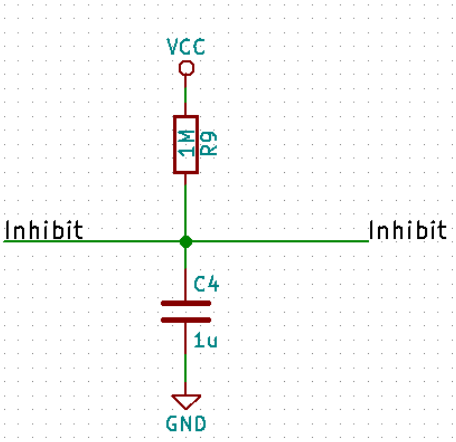


Fig. 3.58: Reset del sistema en modo manual

A la hora de conectar todo el sistema, se ha observado que al dejar el Inhibit del pedal en circuito abierto éste se activaba, por lo que no se puede mantener el modo automático, ya que uno de los indicadores nos estaba devolviendo continuamente al modo manual, por lo que se ha realizado un sistema pull-up con una resistencia (Fig. 3.59), evitando así este falso positivo.

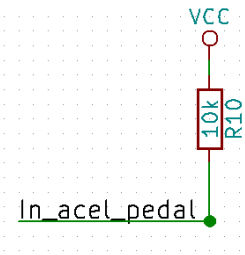


Fig. 3.59: Pull-up para los Inhibits

El led del pulsador puede ser encendido de la forma en que está el circuito realizado, pero no es la manera óptima, ya que el CD4093BPW no es capaz de dar la corriente suficiente que demanda el led, por lo que cae la tensión de la salida de éste, pudiendo ocasionar problemas en un futuro.

Por ello, se ha añadido un DG419DY que proporcione la corriente que demanda el led sin que ocurra una caída de tensión en la salida de la puerta.

3.6 Segunda versión de la placa de conmutación

Debido a que la placa de circuito impreso es el primer prototipo será necesaria la implementación de una segunda placa que contenga más sistemas de control.
En la segunda versión se añadirán las modificaciones hechas en la anterior versión, además de implementar la etapa de potencia y el microcontrolador. De esta manera, todo estará unificado en una única placa con unas dimensiones concretas diseñada para esta operación.

Para la implementación del pedal del freno se va a utilizar una señal que actualmente proporciona el vehículo, pero sería necesario cambiarla de cara a hacer un freno automatizado, ya que esta señal también se activaría cuando el freno se accionara de manera automática.

Para la implementación del Inhibit del volante es necesario utilizar las señales recibidas del sensor de torque de la servodirección, que diferencian entre las fuerzas aplicadas desde el motor y las aplicadas de forma manual en el volante por el conductor. Con ambas señales y su correcto procesamiento mediante el uso de comparadores se podrá implementar la señal de Inhibit del volante.

Capítulo 4: Conclusiones

Este trabajo se ha basado en la compresión de un sistema externo, como es el vehículo eléctrico adquirido para el proyecto SmartElderlyCar, para poder realizar así las modificaciones necesarias y lograr la automatización del vehículo, así como proporcionar un sistema drive-by-wire.

Se ha profundizado en la electrónica más básica adquirida a lo largo de los cursos del Grado, culminando con el diseño y fabricación de dos placas de circuito impreso, algo que nunca se había realizado anteriormente en la carrera, a pesar de ser imprescindible para la formación de un Ingeniero Electrónico.

Además, para el desarrollo de la dirección y el acelerador, se han podido mezclar actuadores de fácil acceso en el mercado como son la servodirección de un vehículo comercial, así como un DAC para abordar esta parte del diseño, todo ello controlado desde un micro controlador.

Los resultados obtenidos, como en todas las primeras versiones, son mejorables, aunque la funcionalidad la cumple a la perfección. Algunas de las tareas que se podrían mejorar serían la comunicación con el sistema de control mediante el bus CAN del micro controlador, que es el estándar de la gran mayoría de vehículos, además de mejorar el control del acelerador y la dirección añadiendo un PID. Por otra parte, en la placa de circuito impreso se debería generar una segunda versión que contenga la etapa de potencia de la servodirección con el Driver, y todos los elementos del micro controlador, además de las modificaciones que ha sido necesario realizar. De esta manera reduciríamos el espacio y los elementos distintos en el conjunto del vehículo.

Como añadido, este TFG ha permitido al autor trabajar dentro de un grupo de investigación en un proyecto complejo y real, siendo necesaria la interacción de varias personas para llevar a cabo las diferentes tareas a realizar.

Anexo I: Pliego de condiciones

Requisitos

Hardware

- Ordenador Acer Aspire V3-572G-56ND
  - Procesador Intel Core i5-5200U
  - Memoria RAM de 8 GB DDR3
  - Disco duro SSD 250 GB
  - Disco duro HDD 1000 GB
  - Tarjeta gráfica NVIDIA GeForce 840M con 2 GB de VRAM dedicada
- Instrumentación electrónica
  - Multímetro
  - Osciloscopio
  - Fuente de alimentación de corriente continua
- Estación de soldadura JBC
- Microscopio
- Pinzas
- Calibre
- Crimpadora

Software

- Sistema operativo Linux distribución Ubuntu 16.04
- Kicad
- Netbeans
- Arduino Uno 1.8.5
- MatLab R2018a
- LibreOffice
- Microsoft Word

Anexo II: Manual de usuario

Para el desarrollo de ambas partes, es necesario utilizar cierto software especializado.

Conexionado del sistema

En el capítulo 2, se ha explicado el funcionamiento de los sistemas implementados en el vehículo para el control del acelerador y la velocidad.

Para que funcione debidamente, es necesario seguir una serie de pasos para su realización conexionado y posterior puesta en marcha.

1. Entendimiento del Hardware existente en el vehículo.
2. Cambio del Hardware, del vehículo para que se ajuste a las necesidades del diseño.
3. Diseño del Software del micro controlador. El software usado es Arduino IDE 1.8.5, que se puede descargar desde la página oficial. Como el micro controlador usado no es exactamente el genérico de Arduino, se debe instalar tanto los controladores de la placa como las librerías que se han utilizado, y añadirlas al proyecto.
4. Implementación del sistema.
5. Conexionado del sistema.
    A. Para el acelerador y encoders, no da posibilidad a error debido al uso de diferentes conectores para su realización, siendo solo un único conexionado, el cual es el correcto.
    B. Para los pulsadores, se debe conectar los dos faston provenientes del vehículo al pulsador de encendido, de esta manera se podrá proceder a la puesta en marcha del vehículo. Y para el pulsador de cambio de modo de conducción, se deben conectar cuatro fastons, dos para el led y dos para el cierre del circuito que genere la señal de cambio de modo.
    C. Para la servodirección, se debe utilizar el conector habilitado para ella, siendo imposible equivocarse ya que solo se dispone de una posición correcta.
    D. Conexón del conector al PCB.
    E. Conexión del Inhibit del freno.

6. Una vez conectado todo, se debe realizar una serie de comprobaciones antes de arrancar el vehículo.
    A. Comprobación de la seta de emergencia.
    B. Comprobar que el cargador no se encuentre conectado.
    C. Establecer el sentido de giro mediante el interruptor.
    D. Quitar el freno de mano.

Kicad

En el capítulo 3, referente al diseño de PCBs, debemos tener en cuenta modo de resumen de la aplicación, que es un software que dispone de varias aplicaciones.

- Para la realización del esquemático usaremos **Eeschema**.
- Para el diseño de huellas, usaremos el **Editor de huellas**.
- Y para la colocación del PCB, usaremos **Pcbnew.**

Y, por último, para el rutado de la placa, debemos acudir al proyecto creado en Netbeans, para el desarrollo del Autorouting.

Anexo III: Presupuesto

A continuación, se desglosan los costes estimados para este proyecto

Costes de material

Para la realización de este trabajo ha sido necesario la utilización de diferentes herramientas para la realización de operaciones de instrumentación, soldadura o creación de cables

Concepto	Precio unidad	Unidades	Subtotal
Ordenador Acer Aspire V3-572G-56ND	643,00 €	1	643,00 €
Multímetro	181,00 €	1	181,00 €
Osciloscopio	396,58 €	1	396,58 €
Fuente de alimentación DC	150,00 €	1	150,00 €
Estación de soldadura JBC	3.775,20 €	1	3.775,20 €
Microscopio	275,00 €	1	275,00 €
Pinzas	6,14 €	1	6,14 €
Calibre	161,42 €	1	161,42 €
Crimpadora	18,79 €	1	18,79 €

Tab. 1: *Costes de material*

Costes de software

Debido a que el proyecto es OpenSource el software utilizado debe ser de libre acceso, es por eso por lo que los principales programas utilizados son de coste 0. Y los utilizados para el análisis y redacción de este trabajo como son Matlab y el paquete Office han sido proporcionados por la Universidad de Alcalá.

Concepto	Coste de licencia
Ubuntu 16.04	0,00 €
Arduino IDE 1.8.5	0,00 €
Kicad	0,00 €
Matlab R2018a	0,00 €
Microsoft Office	0,00 €
Netbeans	0,00 €

Tab. 2: *Costes de software*

Costes de realización de las ECUs

Aquí se engloban los materiales necesarios para la realización de la dirección y el acelerador autónomo.

Concepto	Precio unidad	Unidades	Subtotal
Servodirección	150,00 €	2	300,00 €
Bobina de cable	18,00 €	1	18,00 €
Termoretractil	1,00 €	4	4,00 €
Fabricación sensor de ángulo	17,00 €	5	85,00 €
Componentes sensor de ángulo	9,00 €	2	18,00 €
Caja	5,00 €	1	5,00 €
Olimexino-STM32	19,95 €	5	99,75 €

Tab. 3: *Costes de realización de las ECUs*

Costes de la realización de la placa de conmutación

Los costes aquí desglosados permiten la fabricación de dos protoipos completos, teniendo componentes para una tercera, salvo el conector.

Concepto	Precio unidad	Unidades	Subtotal
Fabricación PCB	15,60 €	5	78,00 €
Componentes PCB	14,67 €	3	44,01 €
Conectores automoción	5,00 €	2	10,00 €

Tab. 4: *Costes de la realización de la placa de conmutación*

Costes de personal

Para los costes de personal se ha desglosado el proyecto las horas del proyecto con un sueldo medio de 15 euros por hora trabajada de un Ingeniero Industrial.

Concepto	Precio por hora	Horas	Subtotal
Entendimiento del Hardware	15,00 €	72	1.080,00 €
Realización de las ECUs	15,00 €	240	3.600,00 €
Realización de la placa de conmutación	15,00 €	192	2.880,00 €

Tab. 5: *Costes de personal*

Costes totales

Concepto	Subtotal
Materiales	5.607,13 €
Software	0,00 €
Realización de las ECUs	529,75 €
Realización de la placa de conmutación	132,01 €
Personal	7.560,00 €
Subtotal Final	**13.828,89 €**

Tab. 6: *Costes totales*

Gastos generales y beneficio industrial

El beneficio industrial y los gastos generales son desembolsos que vienen tipificados por la ubicación de las instalaciones de trabajo e ingresos generados a nivel de manufactura. Se estimará un 15% sobre el coste de ejecución material.

Concepto	Subtotal
Coste de ejecución de material	13.828,89 €
Porcentaje estimado	15%
Subtotal final	2.074,33 €

Tab. 7: *Gastos generales y beneficio industrial*

Presupuesto de ejecución por contrata

Presupuesto igual a la suma de los costes de ejecución y gastos generales y beneficio industrial.

Concepto	Subtotal
Coste de ejecución de material	13.828,89 €
Gastos generales y beneficio industrial	2.074,33 €
Subtotal final	15.903,22 €

Tab. 8: *Presupuesto de ejecución por contrata*

Importe total del presupuesto

Para terminar el presupuesto se debe añadir el IVA (21%) a partir del presupuesto de ejecución por contrata.

Concepto	Subtotal
Presupuesto de ejecución por contrata	15.903,22 €
Porcentaje estimado	21%
Subtotal final	3.339,68 €
Importe final	19.242,90 €

Tab. 9: *Importe total presupuestado*

Bibliografía

[1] Alex Barredo, Coches autónomos, el estado del arte.
https://programarfacil.com/podcast/coche-autonomo-estado-del-arte/#Coche_autonomo_que_es

[2] Bundesanstalt für Straßenwesen
Brüderstraße 53, 2013, "Legal consequences of an increase in vehicle automation"
http://bast.opus.hbz-nrw.de/volltexte/2013/723/pdf/Legal_consequences_of_an_increase_in_vehicle_automation.pdf

[3] Web oficial con las características y link de compra del micro controlador Olimexino-STM32.
https://www.olimex.com/Products/Duino/STM32/OLIMEXINO-STM32/open-source-hardware

[4] Web oficial Kicad
http://kicad-pcb.org/

[5] Web del grupo de investigación Robesafe
https://www.robesafe.uah.es/index.php/es-es/

[6] Web del proyecto SmarElderlyCar
https://www.robesafe.uah.es/index.php/en/smartelderlycar

[7] Bumblebee XB3
https://www.ptgrey.com/bumblebee-xb3-1394b-stereo-vision-camera-systems-2

[8] Filtro de Kalman
https://es.mathworks.com/discovery/filtros-kalman.html

[9] Mapa de Lanelets
https://github.com/phbender/liblanelet

[10] Sistema operativo ROS
http://www.ros.org/

[11] Vehículo Tabby Evo
https://www.openmotors.co/osv-platform/

[12] Manual de uso de FreeRTOS
https://www.freertos.org/index.html

[13] Hoja de características del DAC MCP4725

https://www.sparkfun.com/datasheets/BreakoutBoards/MCP4725.pdf

[14] Librería del DAC MCP4725 para micro controladores basados en Arduino

https://github.com/adafruit/Adafruit_MCP4725/blob/master/Adafruit_MCP4725.h

[15] Driver IBT-2

http://www.hessmer.org/blog/2013/12/28/ibt-2-h-bridge-with-arduino/

[16] Erich Zabler; Klaus Marx; Franz Jost; Abendroth, Marbach; Hans Braun, (1999), "Method and device for angular measurement of a rotatable body"

https://encrypted.google.com/patents/US5930905

[17]Hoja de características del sensor de ángulo KMZ41

https://www.nxp.com/docs/en/data-sheet/KMZ41.pdf

[18] Hoja de características del ADC ADS1115

http://www.ti.com/lit/ds/symlink/ads1115.pdf

[19] Librería del ADC ADS1115 para micro controladores basados en Arduino

https://github.com/adafruit/Adafruit_ADS1X15/blob/master/Adafruit_ADS1015.h

[20] Hoja de características del dispositivo CD4093BPW

http://www.ti.com/lit/ds/symlink/cd4093b.pdf

[21] Funcionamiento disparador de Schmitt

https://es.wikipedia.org/wiki/Disparador_Schmitt

[22] Hoja de características del DG419DY

https://www.vishay.com/docs/70051/dg417.pdf

[23] Condensadores de desacoplo

http://www.learningaboutelectronics.com/Articulos/Condensador-de-desacoplo.php

[24] Pulsador cambio de modo

https://es.rs-online.com/web/p/botones-pulsadores/1115809/

[25] Conector con protecciones diseñado para automoción

http://www.te.com/usa-en/product-966658-1.html

[26] Librería en la que se encuentra la Shield de Arduino

https://github.com/Alarm-Siren/arduino-kicad-library

[27] Hoja de características del conector con protecciones diseñado para automoción

http://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Customer+Drawing%7F966658%7FG1%7Fpdf%7FEnglish%7FENG_CD_966658_G1.pdf%7F966658-1

[28] Empresa de fabricación de PCBs

https://be.eurocircuits.com/shop/analysisUploaddata.aspx

Universidad
de Alcalá